НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр Коваль
«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Геометричне моделювання в інформаційних системах»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»
на тему: «Модуль розробки навчальних планів інформаційної системи автоматизованої підтримки навчальної діяльності кафедри»

Виконав (-ла):
студент (-ка) IV курсу, групи ТР-61
Дмитрук Андрій Валерійович _____

Керівник:
Професор, доктор технічних наук, доцент
Шушура Олексій Миколайович _____

Рецензент:
зав кафедри Інформаційних систем та технологій
Державного університету телекомунікацій,
д.т.н., професор Сторчак К.П. _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Геометричне моделювання в інформаційних системах

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту
Дмитруку Андрію Валерійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи _Модуль розробки навчальних планів інформаційної системи автоматизованої підтримки навчальної діяльності кафедри_____
керівник роботи _Шушура Олексій Миколайович, д.т.н., доцент _____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № **1168-с**
2. Строк подання студентом роботи ___7 червня 2020 року_____
3. Вихідні дані до роботи Тимчасове положення про організацію освітнього процесу в КПІ ім. Ігоря Сікорського_____
4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
Постановка задачі, аналіз існуючих програм для створення навчальних планів, вибір засобів розробки програмного продукту, розробка моделі бази даних для модулю, створення програмного продукту, робота користувача з програмною системою._____

5. Перелік ілюстративного матеріалу
Постановка задачі, вимоги до системи, діаграма прецедентів доступного функціоналу для користувача, вибір засобів розробки, модель бази даних, шаблон трьохрівневої архітектури системи, діаграма класів моделі даних, діаграма класів контролера, діаграма класів представлення, авторизації в системі, форма створення нового

навчального плану, форма створення плану освітнього процесу, перегляд інформації про обраний навчальний план, висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання "13" квітня 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	13.04.2020	
2.	Вивчення та аналіз задачі	20.04.2020	
3.	Розробка архітектури та загальної структури системи	29.04.2020	
4.	Розробка структур окремих підсистем	06.05.2020	
5.	Програмна реалізація системи	17.05.2020	
6.	Оформлення пояснювальної записки	02.06.2020	
7.	Захист програмного продукту	10.06.2020	
8.	Передзахист	10.06.2020	
9.	Захист	17.06.2020	

Студент _____ Дмитрук А.В.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Шушура О.М.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою даної дипломної роботи є модуль розробки навчальних планів інформаційної системи автоматизованої підтримки навчальної діяльності кафедри. Розроблений модуль автоматизації надає можливість вводити та видаляти базові дані, створювати та переглядати інформацію про навчальний план. Розроблений модуль дозволяє автоматизувати створення навчальних планів кафедри.

Дипломну роботу виконано на 87 аркушах, вона містить 56 ілюстрацій, 18 посилань на літературу та 4 додатки.

ABSTRACT

The purpose of this thesis is a module of curriculum development of the information system of automated support of educational activities of the department. The developed automation module provides an opportunity to enter and delete basic data, create and view information about the curriculum. The developed module allows to automate the creation of curricula of the department.

The thesis is completed on 87 sheets, it contains 56 illustrations, 18 references to literature and 4 annexes.

ЗМІСТ

ВСТУП

Стрімкий розвиток комп'ютерної техніки і її різноманітного програмного забезпечення – це одна з характерних прикмет сучасного періоду розвитку суспільства. Технології, основним компонентом яких є комп'ютер, проникають практично в усі сфери людської діяльності. Висуваються нові вимоги до змісту вищої освіти. Спеціаліст кожного нового випуску того чи іншого навчального закладу завжди повинен мати більш високий рівень підготовки, ніж фахівець попереднього випуску. Якість підготовки фахівця багато в чому визначається програмою його навчання, і, зокрема, головним документом цієї програми – навчальним планом вузу.

Практично кожен викладач у повсякденній роботі зустрічається з різними наборами документів, такими як, наприклад, програми, рейтинги, документи на переобрання і на отримання нового звання. А частина викладачів, окрім інших документів, відповідальні ще й за складання навчальної документації – навчальних планів, робочих навчальних планів, розподілу навантаження викладачів і навчальних аудиторій та інших документів. У всіх цих документів є велика кількість спільних даних, тому розробляти окремі бази даних для кожного напряму документації є недоцільним, оскільки довелося б заповнити велику кількість таблиць-словників, що зберігаються у різних місцях, однією і тією ж інформацією. Це привело б до великих затрат часу, дискового простору (матеріальні затрати), а також збільшувало б ймовірність виникнення помилок при заповненні, яких у офіційних документах не повинно бути у жодному разі.

Для пошуку і отримання потрібної інформації, формування навчальних та робочих навчальних планів, підрахунку потрібних елементів (кількість годин СРС та аудиторних занять тощо) самостійно завжди витрачається велика кількість зусиль та часу, крім того, можна відмітити, що кількість документів постійно зростає, наявні документи часто змінюються, тому постійно доводиться вносити корективи у відповідну документацію.

Для уникнення подібних проблем, а також для зручності, пропонується створення єдиної системи документообігу кафедри, що міститиме у собі базу даних, систему її заповнення і систему формування звітів та навчальних документів. У зв’язку з цим дані системи мають наступні завдання:

– інтеграція даних в електронну систему;
– можливість обміну інформації між відділами;
– ефективне збереження даних у базі даних;
– перехід на електронний документообіг;
– автоматизоване створення навчальних планів кафедри;
– забезпечення захисту конфіденційної інформації;
– можливість формувати та скачувати документи Exel.

Актуальність даної теми дослідження зумовлена стрімким інформаційним і технологічним прогресом, в даний час все більше і більше з'являється різних методик і технологій навчання. В даний час процес складання навчальних планів, заснований на досвіді і інтуїції працівників вищих навчальних закладів, потребує серйозного удосконалення. Це особливо актуально в умовах все зростаючих вимог до підготовки фахівців, необхідності частого поновлення навчальних планів, необхідність підвищення якості навчального процесу.

1 ПОСТАНОВКА ЗАДАЧІ

За роки існування кафедри кількість документів, методичних матеріалів та інших ресурсів зростає. Тому актуальною є розробка інформаційної системи автоматизованої підтримки навчальної діяльності кафедри. Одним з основних елементів навчальної діяльності є розробка та аналіз виконання навчального плану. Тому метою цієї дипломної роботи є створення модулю розробки навчальних планів, як складової частини інформаційної системи кафедри.

Для досягнення поставленої мети необхідно вирішити задачі:

- аналіз предметної області;
- аналіз вже існуючих програмних рішень для розробки навчальних планів;
- вибір методів і технологій для розробки;
- створення моделі даних для системи;
- розробка програмного продукту;
- опис взаємодії користувача із системою.

Вимоги до системи:

- зручний та зрозумілий інтерфейс;
- можливість внесення всієї необхідної інформації для створення навчальних планів;
- створення та заповнення навчальних планів;
- можливість перегляду всієї інформації.

2 АНАЛІЗ ІСНУЮЧИХ ПРОГРАМ ДЛЯ СТВОРЕННЯ НАВЧАЛЬНИХ ПЛАНІВ

У сучасному світі обсяг інформації щорічно подвоюється, швидкість ведення бізнесу – збільшується. Щоб бути успішними, різним сучасним організаціям доводиться не просто оперувати великими обсягами даних і документів, а оперувати ними швидко і ефективно. Для створення системи, що задовольняє таким вимогам, необхідно брати до уваги багато чинників.

Навчальний план є нормативним документом університету, який визначає зміст навчання та регламентує організацію навчального процесу зі спеціальності (спеціалізації). Розробка навчальних і робочих навчальних планів є одним з найбільш відповідальних видів методичної роботи науково-педагогічних працівників. Навчальні і робочі навчальні плани розробляються робочими групами випускових кафедр із залученням, за необхідністю, представників інших кафедр.

Розроблення навчальних і робочих навчальних планів є одним з найбільш відповідальних видів методичної роботи науково-педагогічних працівників університету.

Вивчення наявних систем для автоматизації діяльності кафедри керування показало, що на даний момент немає універсальних програмних продуктів, для вирішення поставленої задачі. Наявні ж системи дорогі у вартості і не підходять для поставлених цілей університету.

2.1 Аналіз навчального та робочого плану кафедри

Навчальний план – основний нормативний документ закладу освіти, за допомогою якого здійснюється організація навчального процесу. Навчальний план за бакалаврським напрямом (із зазначенням певної спеціальності) складаються за відповідними формами. Навчальний план містить у собі розподіл залікових кредитів

між дисциплінами, графік навчального процесу, а також план навчального процесу за семестрами, який визначає перелік та обсяг вивчення навчальних дисциплін, форми проведення навчальних занять та їх обсяг, форми проведення поточного та підсумкового контролю, державної атестації.

Навчальний план є нормативним документом навчального закладу, який визначає зміст навчання та регламентує організацію навчального процесу зі спеціальності (напряму підготовки). Навчальні та робочі навчальні плани складаються окремо для кожного освітньо-кваліфікаційного рівня і за кожною формою навчання (у тому числі навчання зі скороченим терміном навчання, а також для студентів-іноземних громадян).

Навчальний план складається на підставі відповідної освітньо-професійної програми і визначає:

- графік навчального процесу;
- зведений бюджет часу (у тижнях);
- перелік та обсяг нормативних і вибіркових навчальних дисциплін та послідовність їх вивчення;
- види навчальних занять та їх обсяг;
- обсяг часу, передбачений на самостійну роботу студентів;
- форми проведення семестрового контролю;
- види, обсяги і терміни проведення практик;
- форми проведення державної атестації.

Робочі навчальні плани складаються окремо для кожного освітньокваліфікаційного рівня та форми навчання, у тому числі навчання зі скороченим терміном навчання, а також для студентів-іноземних громадян. Робочі навчальні плани ухвалюються Вченою радою відповідного факультету (інституту) та затверджуються першим проректором не пізніше ніж за 4 місяці до початку навчального року.

Графік навчального процесу за різними термінами і формами навчання щорічно розробляється навчальним відділом університету і затверджується першим проректором. План навчального процесу обов’язково має включати відомості про[1]:

- нормативну частину програми відповідно до нормативної частини змісту освітньо-професійної програми;
- вибіркову частину програми (самостійного вибору ВНЗ та вільного вибору студентів) відповідно до варіативної складової ОПП;
- шифри та назви нормативних і варіативних навчальних дисциплін відповідно до ОПП;
- шифри та назви практик;
- дипломне проектування;
- кількість кредитів ECTS та годин з кожної навчальної дисципліни;
- розподіл загального обсягу годин на аудиторні (лекції, практичні або семінарські заняття, лабораторні роботи або комп'ютерні практикуми) та самостійну роботу студентів;
- семестровий контроль (екзамени, заліки, диференційовані заліки);
- курсові проекти (роботи);
- кількість аудиторних годин по курсах і семестрах;
- загальну кількість годин тижневого аудиторного навантаження;
- кількість екзаменів;
- кількість заліків;
- кількість курсових проектів (робіт);

2.2 Огляд існуючих систем для університетів

Існує велика кількість програм автоматизованого створення розкладу для університету. Всі вони мають певні переваги і недоліки. Розглянемо найбільш популярні з них та проведемо порівняння.

Для складання навчальних планів у КПІ використовується програма АСПНП[2] – автоматизована система планування навчального процесу, є допоміжною програмою, що призначена для автоматизації робочих місць методистів кафедр і факультетів, розробки та заповнення робочих навчальних планів відповідно до вимог і стандартів МОН.

Програма включає наступні розділи:

- факультет;
- кафедра;
- сервіс;
- допомога;
- вікна;
- вихід.

У програмі АСПНП ми маємо можливість передивитися загалом весь навчальний план за весь термін навчання відповідного рівня кваліфікації. Можна передавати дані в електронному вигляді до системи «Деканат» та Інформаційно-обчислюваного центру забезпечення навчального процесу. Для цього необхідно експортувати базу даних кафедри. Всі електронні документи можна роздрукувати або зберегти дані на сервері.

Мінусами даної системи є те, що вона не включає загальний документообіг кафедри, може тільки співпрацювати з системою “Деканат”.

Комп'ютерна програма “КУРС: ВНЗ” призначена для ведення єдиної бази даних ВНЗ, керування освітніми процесами, розрахунку навантажень викладачів, складання розкладу занять, обліку студентів, автоматичного (натисканням однієї кнопки) складання обов'язкових звітів ЗНЗ-1, 77-РВК та 83-РВК, ведення електронних журналів.

Програма “КУРС: ВНЗ” здатна враховувати відомості про інфраструктуру університету (корпуси, поверхи, кабінети, класні кімнати і т.ін.), адміністрацію, викладацький склад, студентів, їх батьків або опікунів, дисципліни, що читаються (предмети); навчальний план, навчальні програми з окремих предметів, відомості про кабінети, їх кількість, розподіл по змінам; встановлювати розпорядок роботи на кожен день, а також протягом тижня, місяця, року. За допомогою програми можна скласти розклад занять – найважливіший документ планування навчального процесу і основний організаційний документ, який визначає роботу викладачів та студентів, адміністрації та всього закладу в цілому і, як наслідок, поліпшується ефективність роботи університету, складаються більш комфортні умови для її плодотворної

роботи. Система має модульну структуру, що дозволяє використовувати як весь її функціонал, так і окремо необхідні модулі. Вікно інтерфейсу зображено на рисунку 2.1.

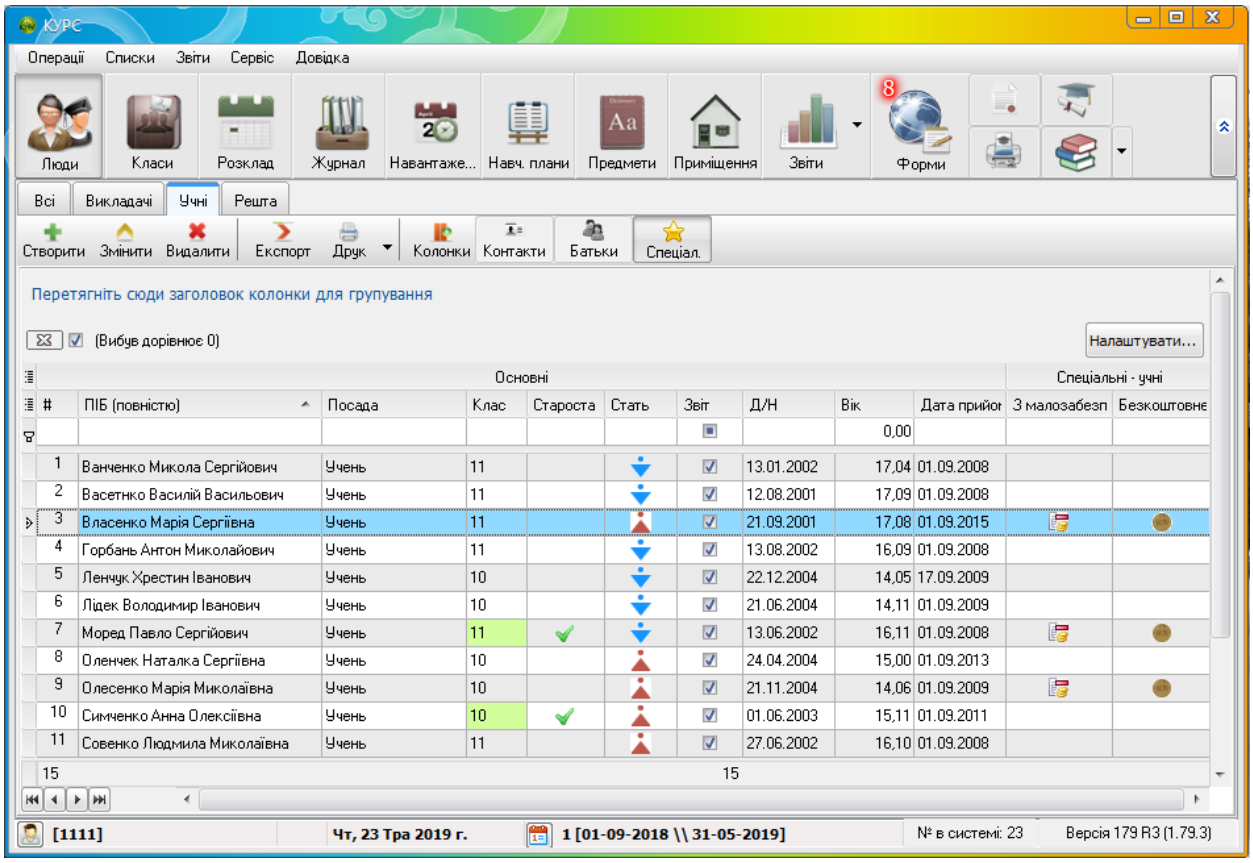

Рисунок 2.1 – Інтерфейс системи “КУРС: ВНЗ”

Система має обмежений функціонал, високі вимоги для використання, платна.

Система “Rector 3”[3] дозволяє створювати навчальний план, має наступні можливості:

– дозволяє складати розклад як у автоматичному, напівавтоматичному так і в ручному режимі;
– при складанні розкладу програма підбере кабінети, враховуючи їх вміст та порівнявши його із кількістю студентів в кабінеті;
– при необхідності програма розділить групу на 2-6 підгруп та розведе їх в різні кабінети до різних викладачів;
– можливе зведення разом декілька груп;
– гнучкі завдання для спарювання пар;

– настройки параметрів дозволять користувачеві управляти якістю розкладу: відслідковувати санітарні норми під час складання розкладу, вести порівняння розкладу з навчальним планом та вказівка на розбіжності, перевірка розкладу на відповідність всім вимогам;
– максимальна зручність при введенні навантажень кожного викладача.

Вікно інтерфейсу зображено на рисунку 2.2.

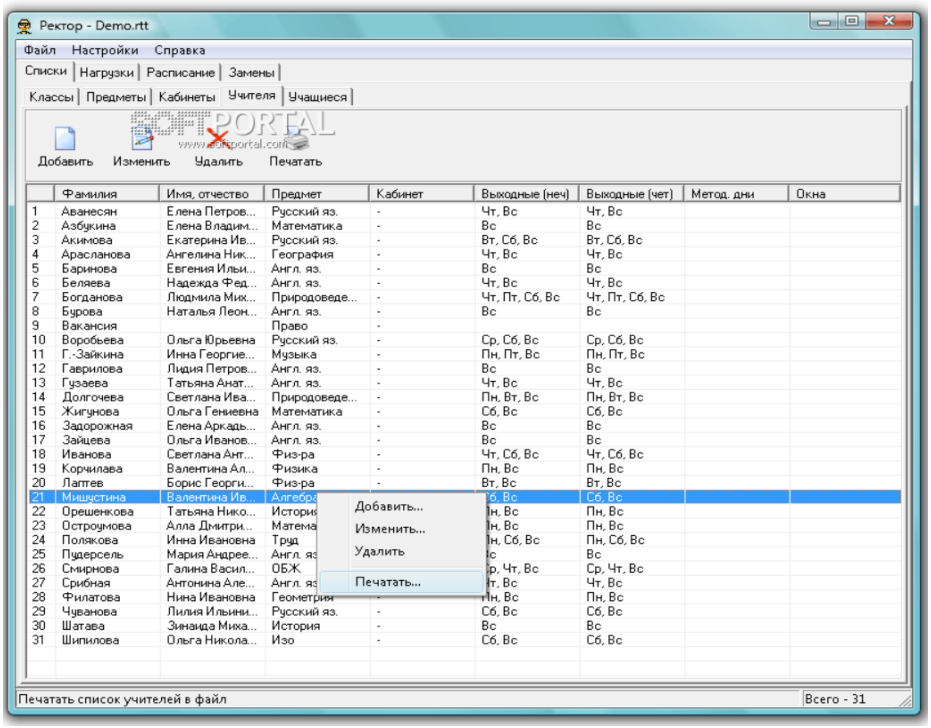

Рисунок 2.2 – Інтерфейс системи “Rector 3”

“aSc TimeTables” – це програма, яка дає вам шанс легко і швидко створювати розкладу з аудиторіями і викладачами, з огляду на тижневі ліміти часу виділені кожному з викладачів. У програмі передбачені всі типи спеціальних розподілів, такі як додавання декількох викладачів до одного і того ж предмету.

Програма може робити окрему конфігурацію аудиторій і викладачів, повідомляючи вас, чи є можливість поставити предмет вищої математики на певну годину, або наприклад, чи вільний певний викладач вранці у вівторок.

Після того як ви ввели вашу інформацію, TimeTables всього за кілька хвилин генерує закінчений розклад. Програма пропорційно розподілить навантаження по всьому тижню, контролюючи повні і роздільні предмети.

TimeTables також може контролювати втручання в розклад, і допоможе вам уникнути типових помилок, дозволяючи вам вносити зміни вручну, але попереджаючи, якщо ви робите помилку.

З мінусів системи, є те що вона платна і має незручний інтерфейс, локалізована тільки англійською. На рисунку 2.3 зображено головний інтерфейс програми.

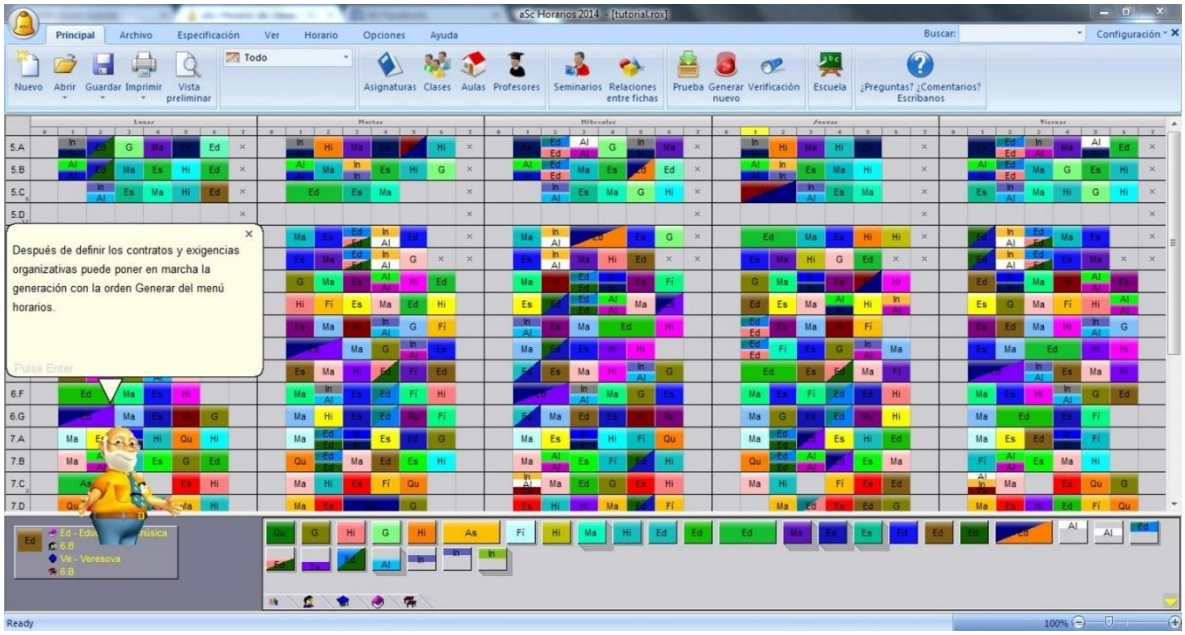

Рисунок 2.3 – Інтерфейс системи “aSc TimeTables”

2.3 Принципи автоматизації навчальної діяльності

Оглянувши та розібравши вище вказані наявні системи, виділено основні принципи в їх побудові та функціонуванні.

Визначаються такі основні принципи автоматизації навчальної діяльності:

- методичну документацію (інформацію про дисципліни, методичне забезпечення предмета, практики, дипломне проектування і т. і.);
- звітну документацію, що стосується викладачів (індивідуальні плани, рейтинги, терміни укладених контрактів і т. і.);
- наукові звіти (щорічний звіт кафедри, форма на обрання та переобрання на посаду тощо);

– навчальні та робочі навчальні плани та іншу документацію, пов’язану з забезпеченням навчального процесу;
– розподіл навантаження викладачів.

Перший етап аналізу методичної системи – визначити цілі і задачі, сформулювати основні проблеми, що потребують розв’язання, виявити основних учасників, визначити їх ролі і методи взаємодії. На другому етапі потрібно знайти шляхи розв’язання ключових проблем та запропонувати відповідні удосконалення системи.

Однією з проблем, є проблема створення навчального і робочого плану, оскільки в цьому питанні потрібно враховувати багато особливостей майбутньої системи.

2.4 Висновки

В даному розділі було проаналізовано документи необхідні для ведення навчального процесу в НТУУ «КПІ». Детально розглянуті особливості складення навчального та робочого навчального планів кафедри, форми розрахунку педагогічного навантаження викладачів, зроблені висновки щодо автоматизації їх формування. Визначено основні складові навчального та робочого планів, форми педагогічного навантаження викладачів, що повинні бути віднесені до системи автоматизації документообігу кафедри.

Підсумовано наявні можливості та властивості таких систем. З огляду на наявні проблеми було вирішено розробити систему, яка відповідатиме основним критеріям:

– надійність і безпека;
– сумісність;
– зручність у використанні та адмініструванні;
– модульність.

3 ВИБІР ЗАСОБІВ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

Для розробки програмного продукту було використано наступні засоби:

– середовище Visual Studio 2019;
– СУБД Ms SQL;
– технологія MVC;
– технологія Entity.

Перш за все для доступу до методичних ресурсів необхідно створити базу даних. Саме для цього і використовується СУБД Ms Sql.

За допомогою середовища Visual Studio 2019 мовою C# та з використанням технологій MVC, Entity буде спроектовано та розроблено WEB додаток , що дає змогу переглядати навчальну інформацію, створювати навчальні і робочі плани.

3.1 Середовище розробки Visual Studio 2019

Для розробки програмного продукту необхідно мати текстовий редактор, в якому можна друкувати код програми. Також потрібен компілятор, який би скомпілював набраний в текстовому редакторі код в додаток exe. Крім того потрібен фреймворк .NET для компіляції і виконання програми.

Щоб полегшити написання, а також тестування та налагодження програмного коду використано середовище розробки Visual Studio Community 2019[5].

Середовище Visual Studio 2019 – це інтегроване середовище розробки програмного забезпечення від компанії Microsoft. За допомогою Visual Studio можна розробляти додатки для Windows, iOS, Android та інших популярних платформ. У Visual Studio доступні інструменти не тільки для створення desktop додатків, але і web, наявні мобільні і хмарні інструменти розробки. Є можливість писати код на таких мовах як: C ++, C #, Visual Basic, F #, JavaScript[6], Python, TypeScript. Крім

усього цього вона включає в себе компілятори, конструктори, редактори, відладчики, візуальні представлення Windows Form і інші, а також величезну кількість розширень для різних областей застосування – від PHP до ігор. На рисунку 3.1 зображено інтерфейс Visual Studio 2019.

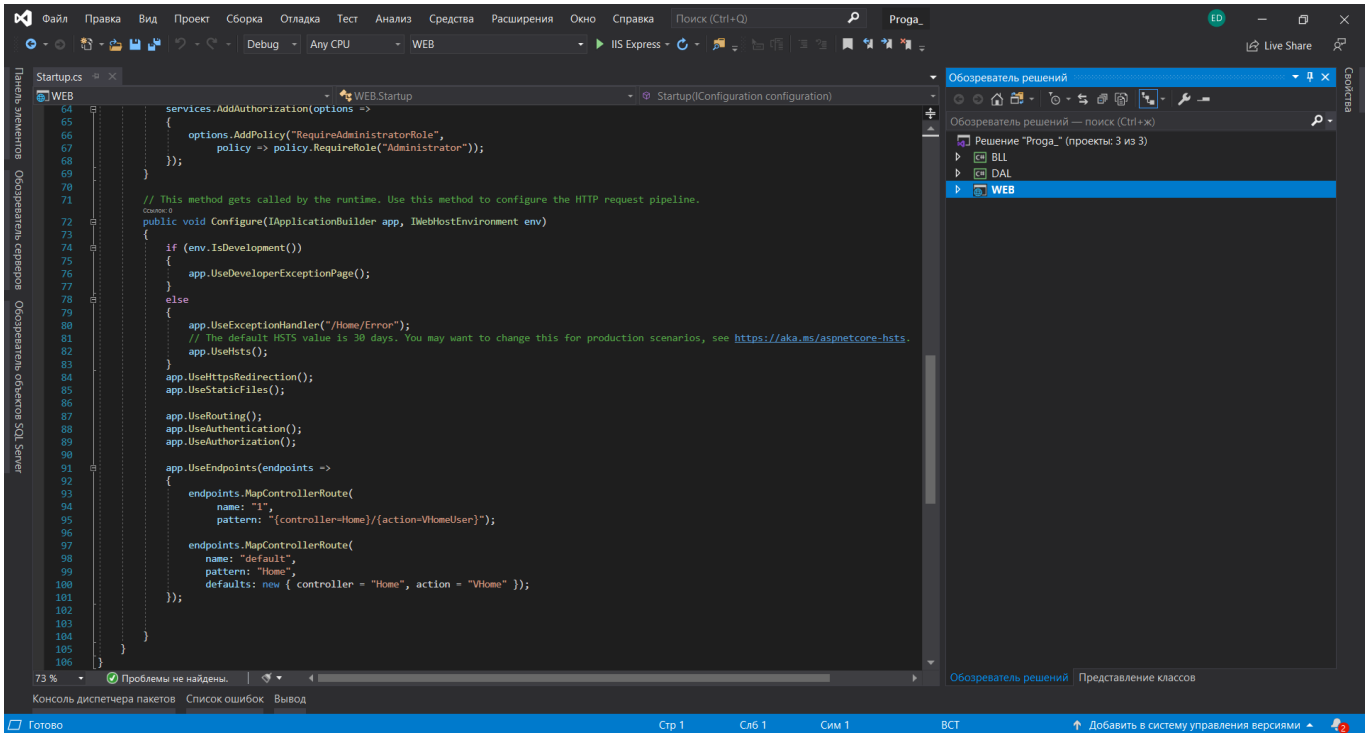

Рисунок 3.1 – інтерфейс Visual Studio 2019

Середовище Visual Studio 2019 року включає в себе наступні нові можливості і поновлення:

– випускається в трьох редакціях (раніше було чотири);
– крос-платформна підтримка мобільних пристроїв (Android, IOS, і Windows);
– покращення в C ++;
– зміни в налагодженні і діагностиці;
– платформа .NET Framework 4.8;
– покращена інтеграція Visual Studio і GitHub;
– платформа Asp.net Core 3.0;
– покращення користувацького інтерфейсу.

3.2 Платформа Asp.net Core 3.0

ASP.NET Core – вільне та відкрите програмне забезпечення каркаса вебзастосунків [7], з продуктивністю вищою ніж у ASP.NET, розроблена корпорацією Microsoft [8] і співтовариством. Це модульна структура, яка працює як на повній платформі .NET Framework, так і на платформі .NET Core.

Фреймворк являє собою повний перепис, який об'єднує раніше окремі ASP.NET MVC та ASP.NET Web API у єдину програмувальну модель.

Не зважаючи на те, що це є новим фреймворком, побудованим на новому веб-стеку, ASP.NET Core має високий ступінь сумісності концепцій з ASP.NET MVC, який об'єднує функціональність MVC, Web API та Web Pages. В попередніх версіях платформи дані технології реалізовані окремо і тому містять багато дублювальної функціональності. Тепер це об'єднано в одну програмну модель ASP.NET Core MVC. Веб-форми повністю вийшли в минуле. Програми ASP.NET Core підтримують програмні версії, в якій різні програми, що працюють на одному комп'ютері, можуть орієнтуватися на різні версії ASP.NET Core. Це не можливо з попередніми версіями ASP.NET Core.

Переваги ASP.NET Core:

- відсутність досвіду розробника (до прикладу, компіляція неперервна, отже розробник не повинен додатково використовувати команду компіляції);
- модульна структура розподіляється як NuGet пакунки;
- cloud-optimized runtime (оптимізована для Інтернету);
- хост-агностик за допомогою Відкритого Веб-Інтерфейсу для .Net (OWIN) підтримки – працює в IIS або в автономному режимі;
- єдина історія для створення веб UI і веб APIs (тобто обидва ті самі);
- система створення конфігурації середовища на основі хмар;
- легкий і модульний HTTP запит;
- створення та запуск крос-платформних додатків ASP.NET Core у Windows, Mac та Linux;

– відкрите джерело та орієнтоване на спільноту;
– Пряме прикріплення версії додатка при націлюванні на .NET Core.

Через вище описані переваги було вибрано використовувати саме ASP.Net Core для поставленої задачі.

3.3 СУБД Ms SQL

MsSQL – вільна система керування реляційними базами даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам.

Зараз MsSQL – одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування. MsSQLнадає багатий набір функціональних можливостей, які підтримують безпечне середовище для зберігання, обслуговування і отримання даних.

MsSQL – характеризується великою швидкістю, стійкістю і простотою використання, була розроблена для підвищення швидкодії обробки великих баз даних.

Вихідні коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатоканальності, що підвищує продуктивність системи в цілому. На рисунку 3.2 зображено інтерфейс MsSQL.

Особливості Ms SQL:

– простота у встановленні та використанні;
– підтримується необмежена кількість користувачів, що одночасно працюють із БД;
– кількість рядків у таблицях може досягати 50 млн.;
– висока швидкість виконання команд;
– наявність простої і ефективної системи безпеки.

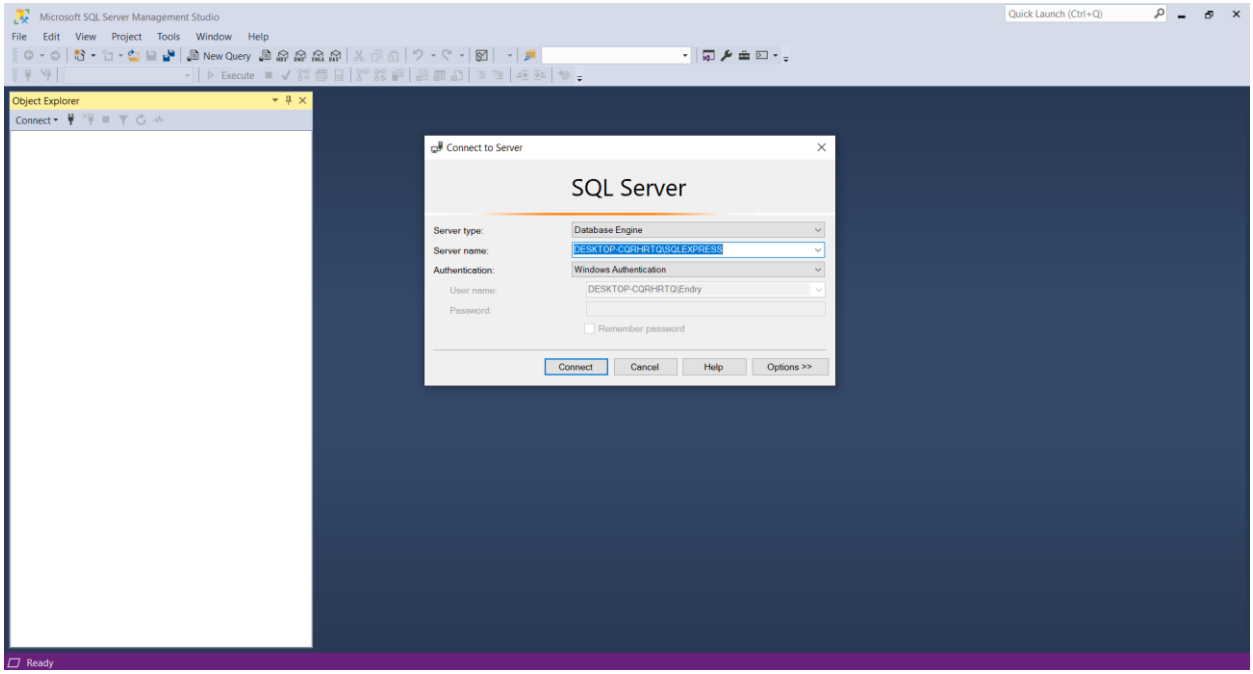

Рисунок 3.2 – Інтерфейс MsSQL

Використано базу даних MsSQL, бо вона має такі переваги над іншими СУБД:

– масштабованість. MsSQL може підтримувати роботу БД значних розмірів, що підтверджують її реалізації в Yahoo!, Google,HP, Associated Press. Згідно документації, що додається до MsSQL, деякі БД, що використовуються компанією MsSQL AB (розробником MsSQL), зберігають до 50 млн. записів;

– переносність. MsSQL працює на різних платформах, серед яких Unix, Linux, Windows, OS/2, Solaris, Mac OC. Окрім того, MsSQL працює на різних платформах;

– зв’язаність. MsSQL має мережеву структуру. До MsSQL можна одержувати доступ із будь-якої точки Internet кільком користувачам одночасно. MsSQL має цілий ряд програмних інтерфейсів додатків (Application Programming Interface –API), які дозволяють встановлювати з’єднання з MsSQL із додатків, написаних на таких мовах як C#, C++, Perl, PHP, Java, Python;

– безпека. MsSQL має систему контроля доступу до даних, забезпечує шифрування даних при передаванні;

- швидкість функціонування;
- зручність експлуатації. MsSQL досить зручно встановлюється та реалізується, легко адмініструється;
- відкритий код.

Мова, що використовується для запитів – Transact-SQL[9], створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO щодо структурованої мови запитів SQL із розширеннями. Використовується як для невеликих і середніх за розміром баз даних, так і для великих баз даних масштабу підприємства. Багато років вдало конкурує з іншими системами керування базами даних.

SQL Server 2005 має вбудовану підтримку .NET Framework. Завдяки цьому, процедури бази даних, що зберігаються, можуть бути написані на будь-якій мові платформи .NET з використанням повного набору бібліотек, доступних для .NET Framework. На відміну від інших процесів, .NET Framework виділяє додаткову пам'ять і будує засоби керування SQL Server, не використовуючи вбудовані засоби Windows. Це підвищує продуктивність порівняно із загальними алгоритмами Windows, оскільки алгоритми розподілу ресурсів спеціально налагоджені для використання у структурах SQL Server.

3.4 Технологія MVC

MVC – архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення [10]. Даний шаблон вибрано оскільки він дозволяє розбити великий проект на менші частини, що полегшує тестування і розширення системи.

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

У рамках архітектурного шаблону модель–вигляд–контролер (MVC) програма поділяється на три окремі, але взаємопов'язані частини з розподілом функцій між компонентами. Модель (Model) відповідає за зберігання даних та їх структуру. Вигляд (View) відповідальний за представлення цих даних користувачеві, тобто інтерфейс програми. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача (зміна положення курсора миші, натискання кнопки, ввід даних в текстове поле) і передає дані у модель. На рисунку 3.3 зображено ідею технології MVC.

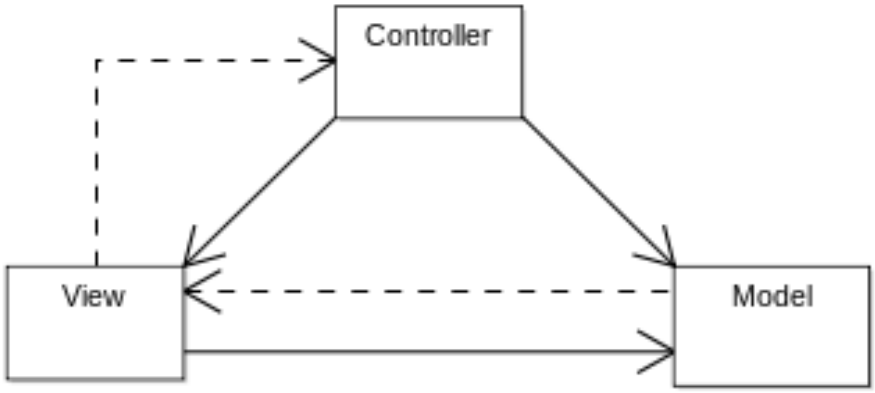

Рисунок 3.3 – Загальна ідеї MVC

Модель є центральним компонентом шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку.

Вигляд може являти собою будь-яке представлення інформації, одержуване на виході, наприклад графік чи діаграму. Одночасно можуть співіснувати кілька виглядів (представлень) однієї і тієї ж інформації, наприклад гістограма для керівництва компанії й таблиці для бухгалтерії.

Контролер одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

Модель інкапсулює ядро даних і основний функціонал їхньої обробки і не залежить від процесу вводу чи виводу даних.

Вигляд може мати декілька взаємопов'язаних областей, наприклад різні таблиці і поля форм, в яких відображаються дані. На рисунку

Мета шаблону – гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

3.5 Технологія Entity Framework Core

Entity Framework Core (EF Core) являє собою об'єктно-орієнтовану технологію від компанії Microsoft для доступу до даних [11]. EF Core є ORM-інструментом (object-relational mapping – відображення даних на реальні об'єкти). Тобто EF Core дозволяє працювати базами даних, але є більш високий рівень абстракції: EF Core дозволяє абстрагуватися від самої бази даних і її таблиць і працювати з даними незалежно від типу сховища. Якщо на фізичному рівні ми оперуємо безпосередньо з таблицями, індексами, первинними і зовнішніми ключами, але на концептуальному рівні, який нам пропонує Entity Framework, ми вже працюємо з об'єктами класів.

Entity Framework Core підтримує безліч різних систем баз даних. Таким чином, ми можемо через EF Core працювати з будь-якої СУБД, якщо для неї є потрібний провайдер.

За замовчуванням на даний момент Microsoft надає ряд вбудованих провайдерів: для роботи з MS SQL Server, для SQLite, для PostgreSQL. Також є провайдери від сторонніх постачальників, наприклад, для MsSQL.

Також варто відзначити, що EF Core надає універсальний API для роботи з даними. І якщо, наприклад, ми вирішимо змінити цільову СУБД, то основні зміни в проекті будуть стосуватися насамперед конфігурації і настройки підключення до відповідних провайдерам. А код, який безпосередньо працює з даними, отримує дані, додає їх в БД і т.д., залишиться колишнім.

Entity Framework Core багато успадкував від своїх попередників, зокрема, Entity Framework 6. У той же час треба розуміти, що EF Core – це не нова версія по відношенню до EF 6, а зовсім інша технологія, хоча в цілому принципи роботи у них

будуть збігатися . Тому в рамках EF Core використовується своя система версій. Поточна версія – 3.0 була випущена у вересні 2019 року. І технологія продовжує розвиватися.

Як технологія доступу до даних Entity Framework Core може використовуватися на різних платформах стека .NET. Це і стандартні платформи типу Windows Forms, консольні додатки, WPF, UWP і ASP.NET Core. При цьому кроссплатформенная природа EF Core дозволяє задіяти її не тільки на ОС Windows, але і на Linux і Mac OS X. Через вище описані переваги, ця технологія буде використовуватися у проекті.

Центральною концепцією Entity Framework є поняття сутності або entity. Сутність визначає набір даних, які пов'язані з певним об'єктом. Тому дана технологія передбачає роботу не з таблицями, а з об'єктами і їх колекціями.

Будь-яка сутність, як і будь-який об'єкт з реального світу, має низку властивостей. Наприклад, якщо сутність описує людини, то ми можемо виділити такі властивості, як ім'я, прізвище, зріст, вік. Властивості необов'язково представляють прості дані типу int або string, але можуть також представляти і більш комплексні типи даних. І у кожної сутності може бути одна або кілька властивостей, які будуть відрізняти цю сутність від інших і будуть унікально визначати цю сутність. Подібні властивості називають ключами.

При цьому суті можуть бути пов'язані асоціативної зв'язком один-до-багатьох, один-до-одного і багато-до-багатьох, подібно до того, як в реальній базі даних відбувається зв'язок через зовнішні ключі.

Відмінною рисою Entity Framework Core, як технології ORM[12], є використання запитів LINQ[13] для вибірки даних з БД. За допомогою LINQ ми можемо створювати різні запити на вибірку об'єктів, в тому числі пов'язаних різними асоціативними зв'язками. А Entity Framework при виконання запиту транслює вираження LINQ в вирази, зрозумілі для конкретної СУБД (як правило, в вирази SQL).

3.6 Висновки

У даному розділі було обрано та описано усі програмні інструменти та їх переваги. Для розробки програмного продукту обрано платформу Asp.Net Core 3.0, мову програмування C#, середовище Visual Studio 2019. Проект буде розроблено з використанням технології Entity та MVC. Обрано реляційну базу даних Ms Sql.

Усі обрані технології є актуальними на даний час, вони є вільним та відкритим програмним забезпеченням, що означає можливість безкоштовного їх використання, навіть для розробки закритих комерційних проектів. Тому продукт буде актуальним і доступним за необхідності, вносити зміни у програмний код самих інструментів для налаштування їхньої роботи на різних системах.

Обраний стек технологій дозволить швидко та якісно розробити майбутній веб-застосунок.

4 РОЗРОБКА МОДЕЛІ БАЗИ ДАНИХ ДЛЯ МОДУЛЮ

Головною задачею системи автоматизованої підтримки навчальних планів кафедри є зберігання інформації необхідної з метою генерації звітів та форм необхідних для функціонування. Існуюча база даних була створена на реляційній моделі даних, використовуючи систему управління базами даних MsSQL[4].

Реляційна база даних (від англ. relation – відношення) – це БД, яка сприймається користувачем як набір нормалізованих відношень різного ступеню. Реляційна модель приховує деталі фізичного зберігання даних від користувача. Уся робота ведеться на логічному рівні, що полегшує виявлення відносин між елементами даних.

За допомогою реляційної БД забезпечено необхідний рівень зв’язку, який дозволить швидко та зручно отримувати доступ до даних, формувати звіти, вести потрібні підрахунки. Ця архітектура передбачає можливість зручного нарощування об’ємів інформації, що важливо, так як БД буде нарощуватися у майбутньому. Також у реляційній моделі досягається інформаційна й структурна незалежність. Записи не зв'язані між собою настільки, щоб зміна однієї з них торкнулася інших, а зміна структури бази даних не обов'язково приводить до перекомпіляції працюючих з нею додатків. Система управління базами даних – комплекс програм, який забезпечує можливість створення, збереження, редагування, пошуку інформації та контролю доступу до БД. На рисунку 4.1 зображено концептуальну модель всієї бази даних.

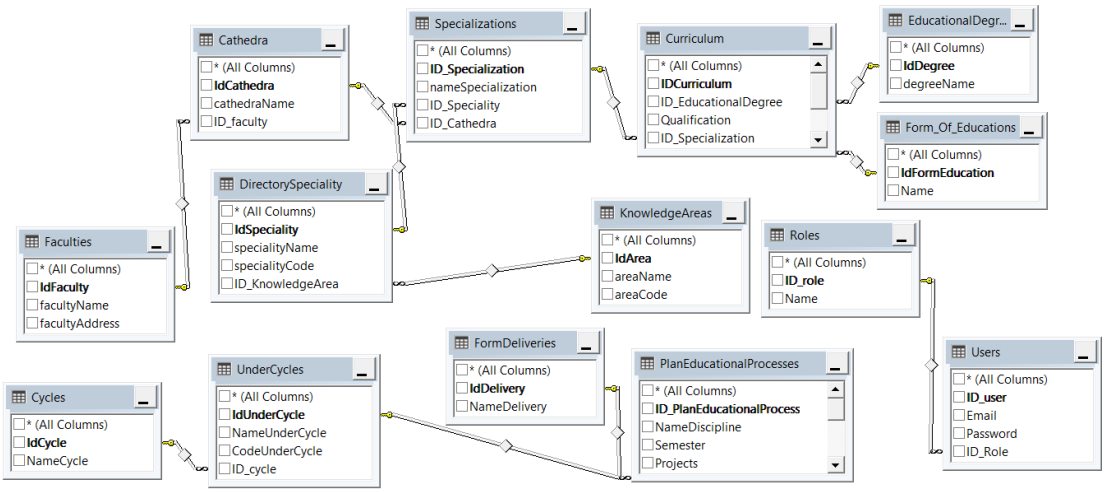

Рисунок 4.1 – Концептуальна модель бази даних

4.1 Проектування сутності “Навчальний план”

Сутність “Навчальний план” є ключовою сутністю підсистеми автоматизації побудови навчальних планів, що відноситься до таблиці сутностей. Саме ця сутність відповідає за автоматизацію створення навчальних планів, адже вона містить основні дані по навчальному плану, а саме: рік навчального плану, освітньо-кваліфікаційний рівень (бакалавр, спеціаліст чи магістр), форму навчання (денна, заочна) та іншу інформацію, що однозначно ідентифікує потрібний навчальний план. На рис. 4.2 зображено сутність "curriculum" (навчальний план), що містить усі поля. Така кількість полів дозволяє зберігати таку кількість інформації, що однозначно визначає потрібний навчальний план.

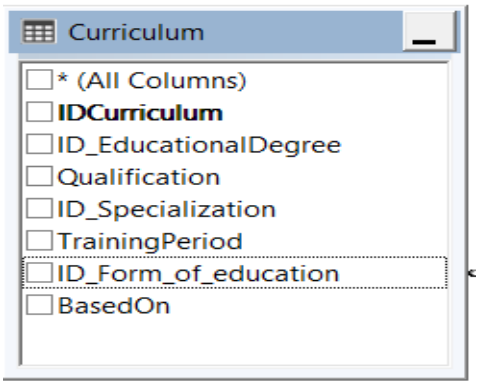

Рисунок 4.2 – Таблиця “Curriculumn” (навчальний план)

Інформація про окремі стовбці таблиці:

- IDCurriculum – оригінальний номер запису в таблиці, який однозначно ідентифікує певний навчальний план. Всі таблиці в БД мають такий стовпець, несучи однакове ідейне навантаження. У всіх випадках полю призначене властивість первинного ключа;
- ID_EducationalDegree – вторинний ключ, що вказує на освітньо кваліфікаційний рівень, що буде здобуто (наприклад, “магістр”"). Ці значення містяться у відповідній таблиці-словнику;
- Qualification – вказує на кваліфікацію, що буде отримана (наприклад, “аналітик комп'ютерних систем”);

- ID_Specialization – вторинний ключ, що вказує на спеціалізацію. Ці значення містяться у відповідній таблиці-словнику;
- TrainingPeriod – поле, що вказує строк навчання у форматі “кількість років кількість місяців” (наприклад, “1 рік 10 місяців”);
- ID_Form_of_education – вторинний ключ, що вказує на форму навчання (денна, заочна, вечірня, екстернат). Ці значення містяться у відповідній таблиці-словнику;
- BasedOn – поле, що вказує, яким освітньокваліфікаційним рівнем володіли студенти до отримання наступного рівня (наприклад, магістра отримують на основі бакалавра, бакалавра на основі повної середньої освіти.

Вибір типу даних для кожного поля (це стосується усіх таблиць БД) проводився з врахуванням максимально-можливої кількості записів при використовуванні підсистеми як “мультипідроздільної”. Наприклад, для первинного ключа встановлений тип “INT”. Всі таблиці бази даних закодовані в UTF-8. В якості строкових типів даних використовується тип “varchar”.

Таблиці-словники, на які посилається “Curriculumn” за допомогою зовнішніх ключів, приведені на рисунках 4.3-4.5.

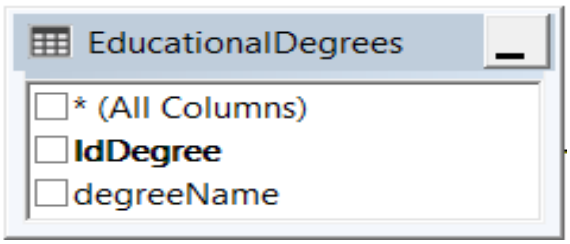

Рисунок 4.3 – Таблиця “EducationalDegrees”

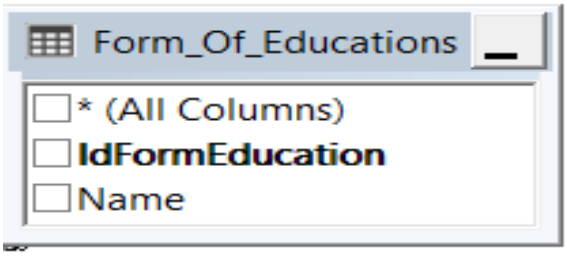

Рисунок 4.4 – Таблиця “Form_Of_Educations”

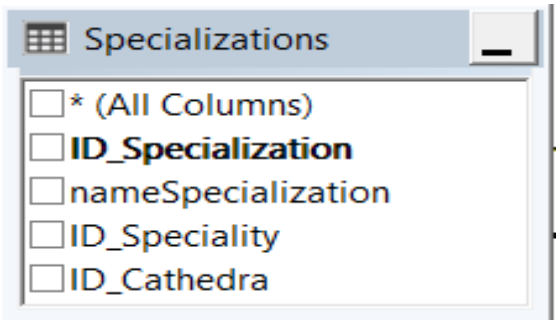

Рисунок 4.5 – Таблиця “Specializations”

Таблиця, зображена на рисунку 4.3, описує кваліфікаційний рівень, який буде здобуто по завершенню навчального періоду. Поле degreeName містить інформацію про назву кваліфікаційного рівня.

Зображена на рисунку 4.4 таблиця описує форму навчання (денна, заочна, вечірня, екстернат). Поле Name містить інформацію про назву форми навчання.

Поля що містяться у таблиці описаній на рисунку 4.5:

– nameSpecialization – поле, що містить інформацію про назву освітньої програми (спеціалізації);
– ID_Speciality – вторинний ключ, що вказує на спеціаліність до якої відноситься дана освітня програма. Ці значення містяться у відповідній таблиці-словнику;
– ID_Cathedra – вторинний ключ, що вказує на кафедру, в якій буде викладатися освітня програма. Ці значення містяться у відповідній таблиці-словнику.

Таблиці на які посилається “Specializations” за допомогою зовнішніх ключів, описано на рисунках 4.6-4.7.

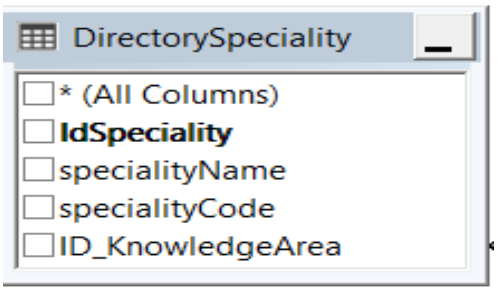

Рисунок 4.6 – Таблиця “DirectorySpeciality”

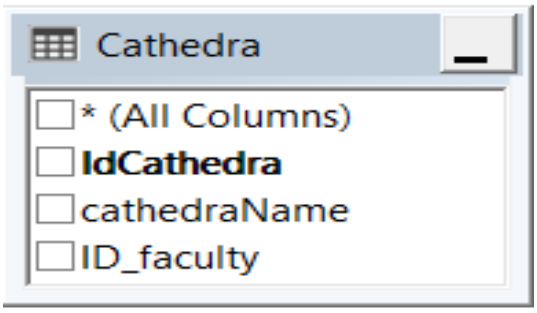

Рисунок 4.7 – Таблиця “Cathedra”

Таблиця зображена на рисунку 4.6 описує інформацію про спеціальність. Поля, що містяться у таблиці:

– specialityName – поле, що містить інформацію про назву спеціальності (спеціалізації);
– specialityCode – поле, містить шифр спеціальності;
– ID_KnowledgeArea – вторинний ключ, що вказує на галузь знань (гуманітарні науки, інформаційні технології) до якої відноситься спеціальність. Ці значення містяться у відповідній таблиці-словнику.

Таблиця, зображена на рисунку 4.7, описує інформацію про кафедру. Містить поля:

– cathedraName – поле, містить інформацію про назву кафедри;
– ID_faculty – вторинний ключ, що вказує на факультет до якого відноситься кафедра. Ці значення містяться у відповідній таблиці-словнику.

Таблиця на яку посилається таблиця “DirectorySpeciality” за допомогою зовнішнього ключа, описано на рисунку 4.8.

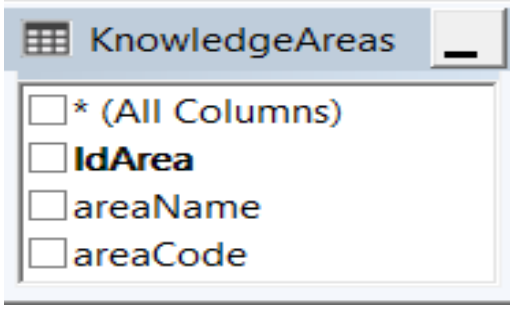

Рисунок 4.8 – Таблиця “KnowledgeAreas”

Таблиця описує:

– areaName – поле, містить інформацію про найменування галузі знань;
– areaCode – поле, містить інформацію про шифр галузі знань.

Таблиця на яку посилається “Cathedra” за допомогою зовнішнього ключа, описано на рисунку 4.9.

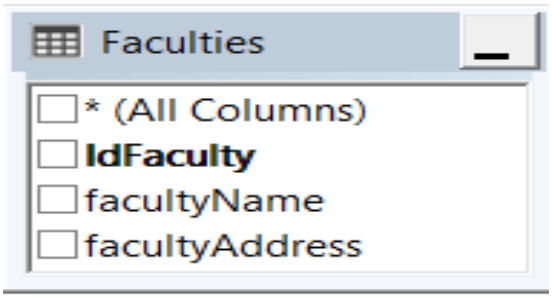

Рисунок 4.9 – Таблиця “Faculties”

Таблиця містить поля:

– facultyName – поле, містить інформацію про назву факультету;
– facultyAddress – поле, містить інформацію про адресу факультету.

4.2 Проектування сутності “План освітнього процесу”

Сутність “План освітнього процесу” є другою головною частиною навчального плану. Ця сутність відповідає за інформацію, яка стосується переліку всіх дисциплін, які будуть викладатись для навчального плану. Таблиця “План освітнього процесу” є однією з основних таблиць, з якими доведеться працювати користувачу БД для заповнення її інформацією по новому начальному плану. Таблиця містить такі важливі поля, як семестр викладання дисципліни, форму здачі предмету, інформацію про наявність курсових робіт та проектів, кількість годин лекцій, практик, годин для самостійного опрацювання та загальний обсяг годин, що виділено на навчальну дисципліну. На рисунку 4.10 зображено таблицю “PlanEducationalProcesses” (план освітнього процесу).

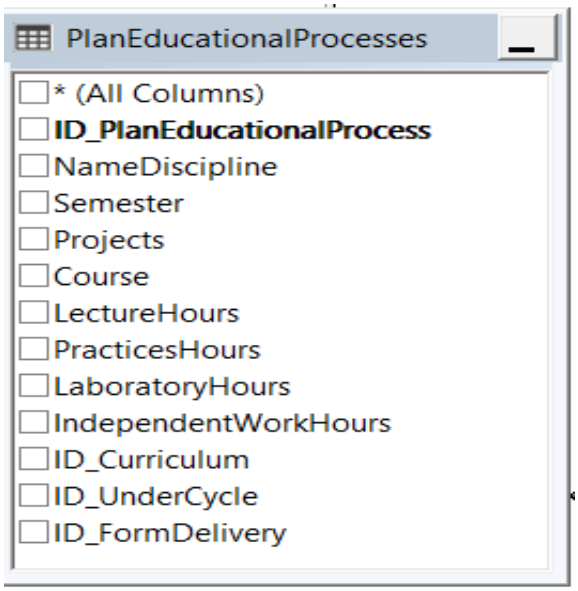

Рисунок 4.10 – Таблиця “PlanEducationalProcesses”

Інформація по окремих стовпцях таблиці:

- ID_ PlanEducationalProcess – оригінальний номер запису в таблиці, який однозначно ідентифікує певний запис у таблиці;
- NameDiscipline – поле, що містить інформацію про назву дисципліни;
- Semestr – поле, що містить номер семестру, в якому буде викладатись дисципліна;
- Projects – поле, що містить інформацію про наявність курсових проектів;
- Course – поле, що містить інформацію про наявність курсових робіт;
- LectureHours – поле, що містить кількість лекційних годин;
- PracticesHours – поле, що містить кількість практичних годин;
- LaboratoryHours – поле, що містить кількість годин, виділених на лабораторні роботи;
- IndependentWorkHours – поле, що містить кількість годин для самостійного вивчення;
- ID_Curriculum – вторинний ключ, що посилається на таблицю Curriculum, яку було описано вище. Цей ключ забезпечує однозначне визначення, для якого саме навчального плану проставляються дані з планом навчального процесу;

– ID_UnderCycle – вторинний ключ, що посилається на таблицю UnderCycle, яка вказує на приналежність певного предмету, певному циклу навчальних дисциплін (наприклад, "Цикл професійної та практичної підготовки"). Ці значення містяться у відповідній таблиці-словнику;
– ID_FormDelivery – вторинний ключ, що посилається на таблицю FormDelivery, яка вказує про тип здачі дисципліни (залік, екзамен). Ці значення містяться у відповідній таблиці-словнику.

Таблиці, на які посилається “PlanEducationalProcesses” за допомогою зовнішніх ключів, приведені на рисунках 4.11-4.12.

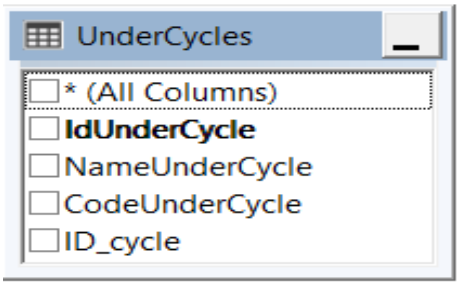

Рисунок 4.11 – Таблиця “UnderCycles”

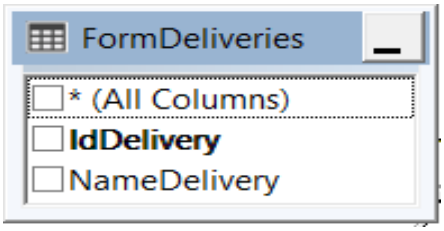

Рисунок 4.12 – Таблиця “FormDelivaries”

Зображена на рисунку 4.11, таблиця постачає інформацію про цикли, на які розбиваються навчальні дисципліни, тобто, кожна навчальна дисципліна належить до якогось циклу (наприклад, предмет "Захист інформації" відноситься до циклу "Дисципліни самостійного вибору навчального закладу").

– NameUnderCycle – поле, що містить назву циклу;
– CodeUnderCycle – поле, що містить шифр циклу;

– ID_Cycle – вторинний ключ, що посилається на таблицю Cycle, яка вказує про тип циклу.

Зображена на рисунку 4.12, таблиця вказує на тип державної атестації, що застосовується до певного предмету. Інформація зберігається у текстовому полі NameDelivery.

Таблиця на яку посилається “UnderCycles” за допомогою зовнішнього ключа, описано на рисунку 4.13.

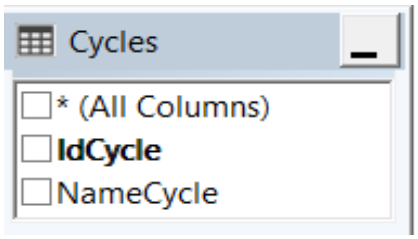

Рисунок 4.13 – Таблиця “Cycle”

Таблиця описує тип циклу (цикл загальної підготовки, цикл професійної підготовки). Інформація зберігається у текстовому полі NameCycle.

4.3 Висновки

В даному розділі було розроблено модель бази даних. Розглянуто всі таблиці і їх поля (типи даних які вони можуть зберігати) з детальним описанням, продемонстровано відношення між всіма таблицями. Дана база даних, призначена для зберігання всієї необхідної інформації для створення навчальних планів кафедри. Оскільки модель бази даних готова, то після цього кроку можна розпочати створення програмного продукту.

5 СТВОРЕННЯ ПРОГРАМНОГО ПРОДУКТУ

Для створення системи було використано середовища: Visual Studio 2019, СУБД MsSQL і дві технології: MVC і Entity. Більша частина дипломної роботи – створення трохрівневої системи. Клієнтський додаток написаний мовою C# на платформі .NET Core у середовищі Visual Studio 2019.

5.1 Створення моделі бази даних

Оскільки у проекті використовується трьохрівнева архітектура, то сам проект розбитий на три частини. Перша частина – Data Access layer (рівень доступу до даних): зберігає моделі, що описують використовувані суті, також тут розміщуються специфічні класи для роботи з різними технологіями доступу до даних, наприклад, клас контексту даних Entity Framework. Тут також зберігаються репозиторії, через які рівень бізнес-логіки взаємодіє з базою даних. Спочатку потрібно інсталювати пакет Entity Framework Core до проекту, для цього використаємо: “Управління пакетами NuGet”. На рисунку 5.1 зображено інтерфейс NuGet.

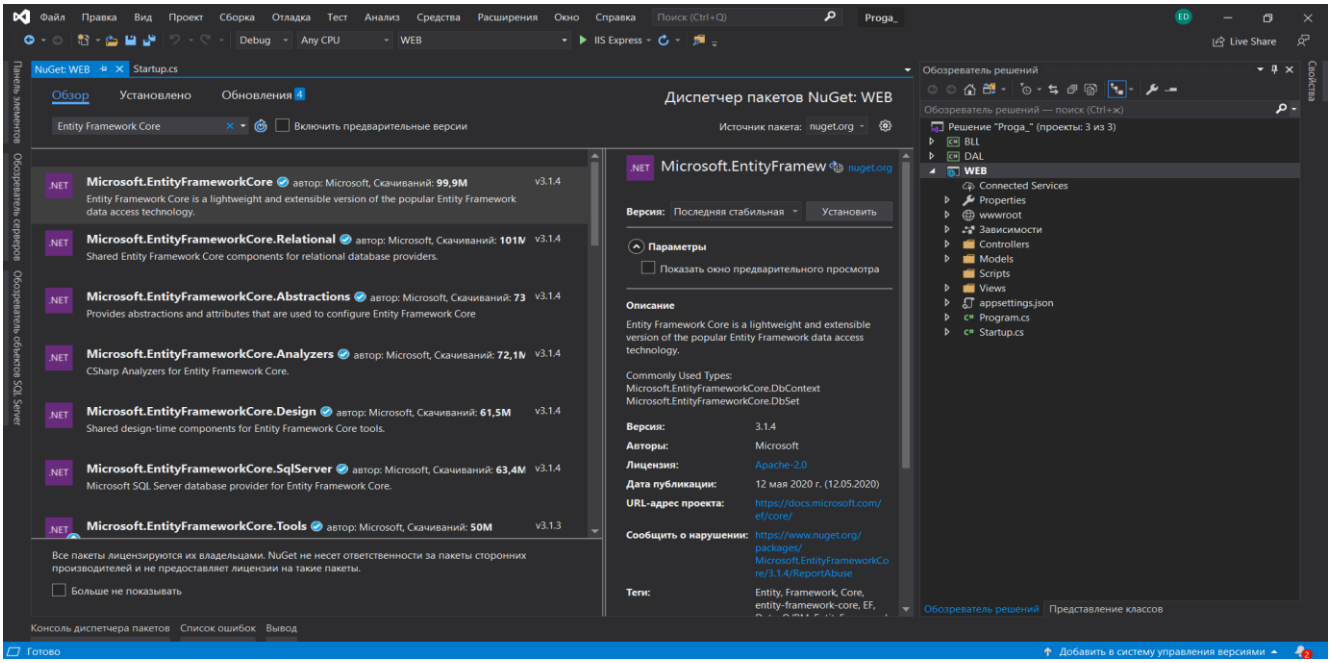

Рисунок 5.1 – Інтерфейс NuGet

Entity Framework підтримує підхід “Code first”[14], який передбачає збереження або вилучення інформації з БД на SQL Server без створення схеми бази даних або використання дизайнера в Visual Studio. Навпаки, створюємо звичайні класи, а Entity Framework вже сам визначає, як і де зберігати об'єкти цих класів (в даному випадку таблиць у базі даних).

Створимо папку Entities, в ній будуть зберігатися класи які будуть виступати таблицями у базі даних, за допомогою технології Entity Framework, підхід code-first. На рисунку 5.2 зображено всі об’єкти-класи для бази даних.

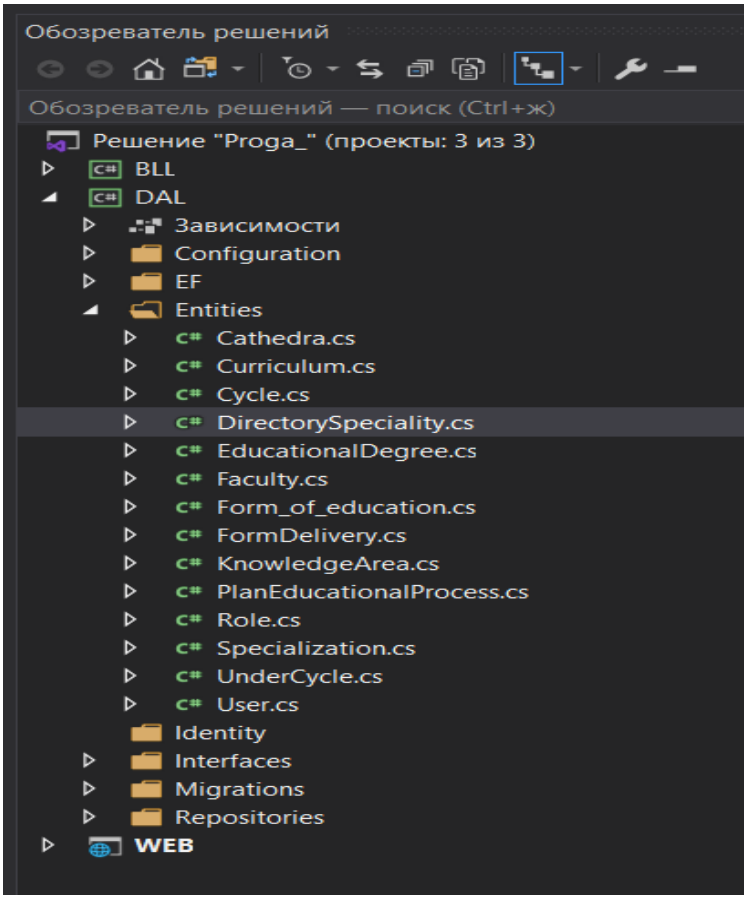

Рисунок 5.2 – Перелік всіх класів-таблиць

Оскільки ми задаємо структуру бази даних, нам відразу потрібно зв’язувати ці класи між собою, для забезпечення відношення між ними. На рисунку 5.3 зображено код класу Curriculumn.

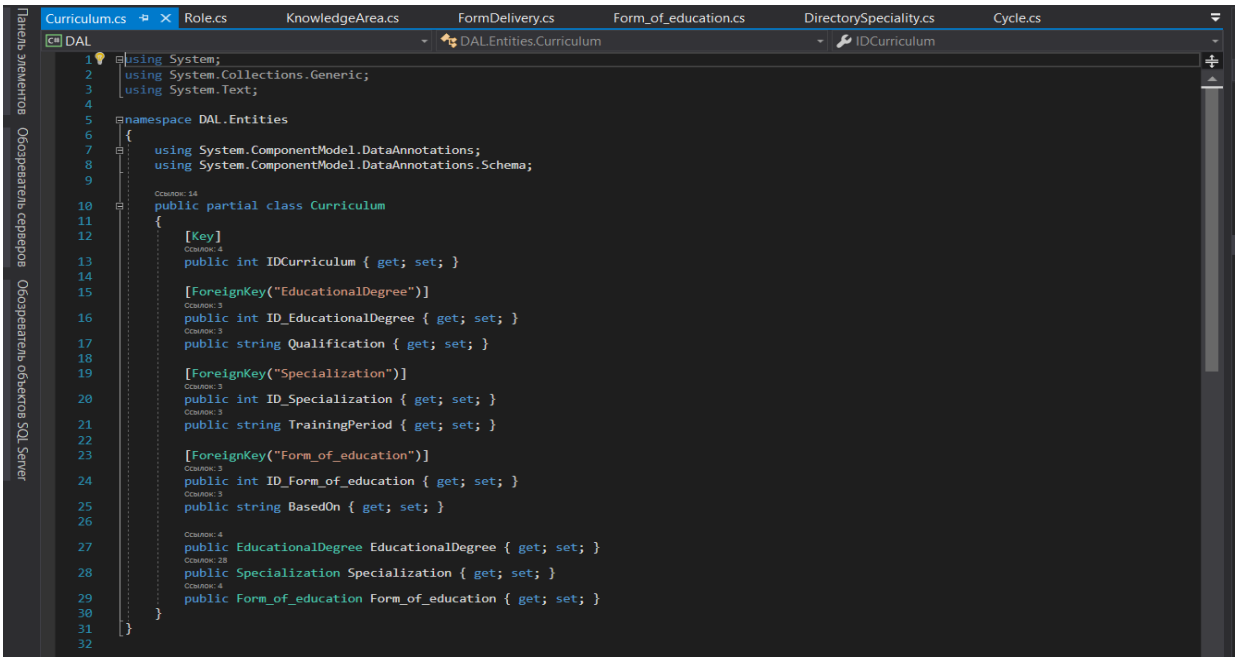

Рисунок 5.3 – Приклад класу Curriculumn

Для створення зв’язків, потрібно до класу підключити бібліотеки:

– using System.ComponentModel.DataAnnotations;
– using System.ComponentModel.DataAnnotations.Schema.

Вони дозволять використовувати записи [Key] і [ForeignKey(“EducationalDegree”)]. Ключ [Key] встановлює до поля в базі даних PrimaryKey. Це означає, що значення в цьому полі повинні бути унікальними і не нульовими, зазвичай для id використовується автоінкремент. Ключ [ForeignKey(“EducationalDegree”)] встановлює зав’язок: “один-до-багатьох” з класом EducationalDegree. І в цьому класі створюється віртуальний об’єкт класу EducationalDegree: public EducationalDegree EducationalDegree { get; set; }

Для підключення до бази даних через Entity Framework, нам потрібен посередник – контекст даних. Контекст даних являє собою клас, похідний від класу DbContext. Контекст даних містить одне або кілька властивостей типу DbSet <T>, де T представляє тип об'єкта, що зберігається в базі даних. Наприклад, створимо контекст даних для роботи з вищенаведеними моделями. На рисунку 5.4 зображено клас ApplicationContext.

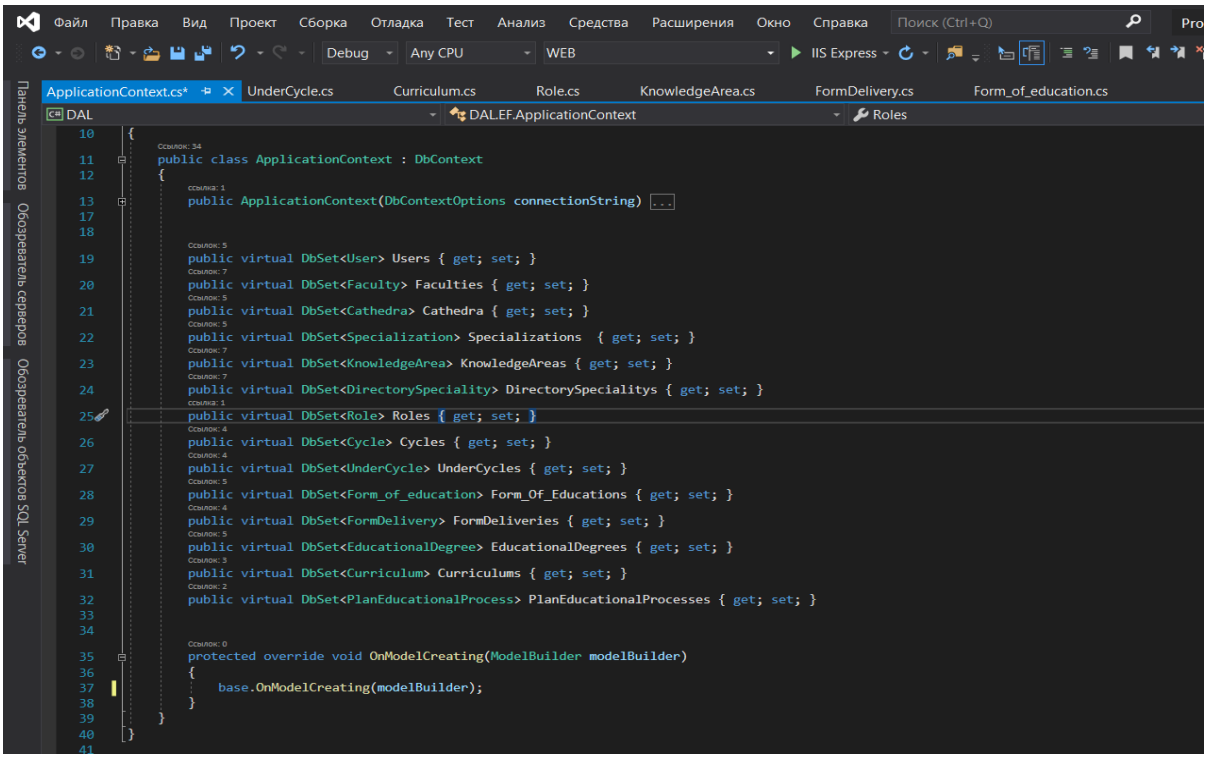

Рисунок 5.4 – Клас ApplicationContext

За допомогою властивостей Users і тд. ми отримуємо доступ до даних відповідних моделей, які зберігаються в базі даних. Конструктор контекста даних приймає строку підключення до бази даних: "Server=DESKTOP-CQRHRTQ\\SQLEXPRESS; Database=DD; Trusted_Connection=True; Max Pool Size=20;Connection Timeout=10".

Для збільшення гнучкості підключення до БД використовуються репозиторії. Тому спочатку визначимо в проекті ще одну папку Interfaces. І в неї додамо інтерфейс репозиторіїв IRepositoryMain. На рисунку 5.5 показаний інтерфейс IRepositoryMain.

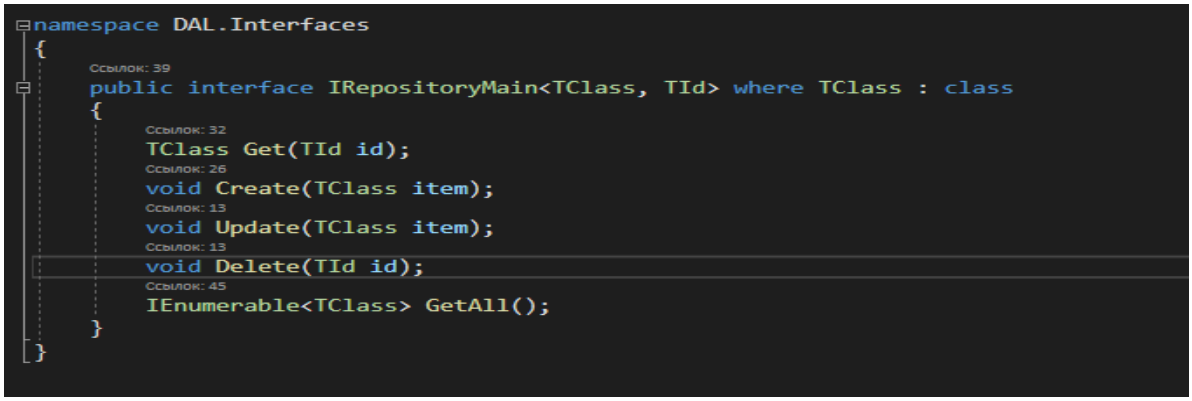

Рисунок 5.5 – Приклад інтерфейсу для репозиторіїв

В цьому інтерфейсі ми вказуємо набір методів, які будуть реалізовуватись у класах-репозиторіях. На рисунку 5.6 зображена реалізація репозиторію для класу Curriculum

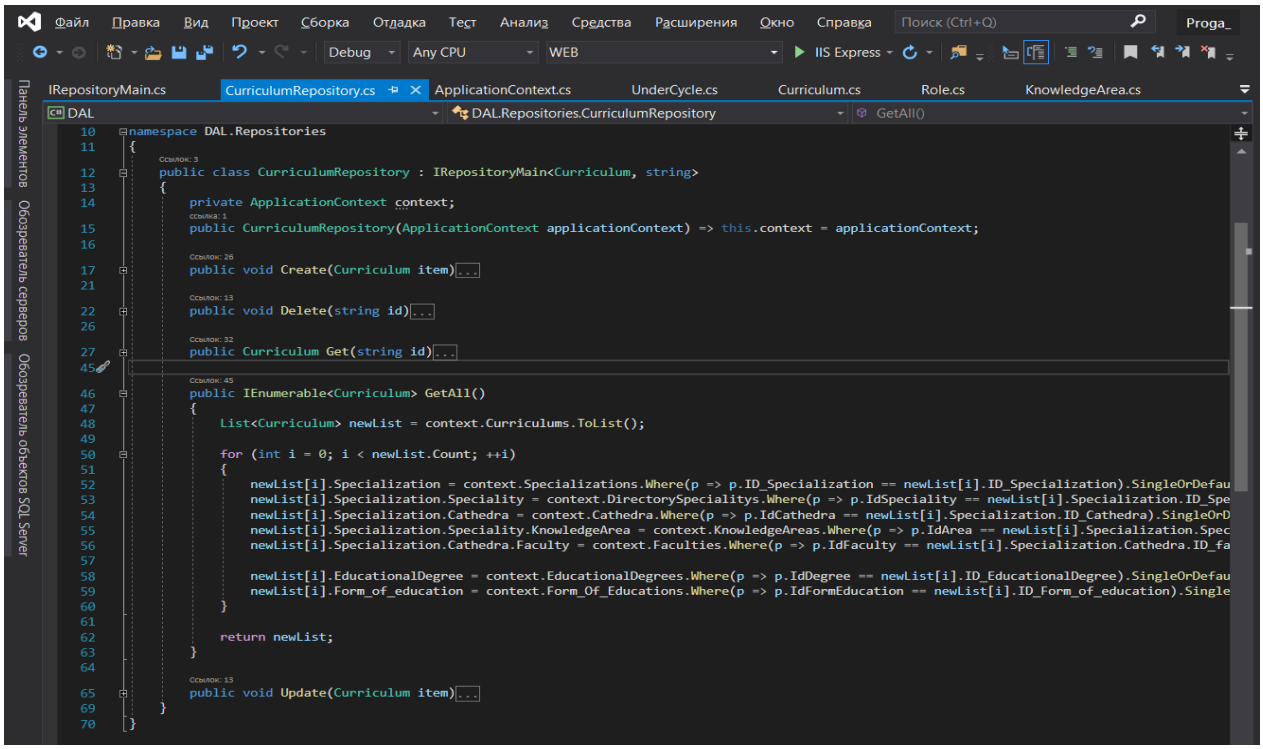

Рисунок 5.6 – Приклад реалізації репозиторію Curriculum

У класі використовується контекст даних ApplicationContext, для роботи з базою даних. За допомогою стандартних методів Entity, ми можемо добавляти, видаляти, змінювати дані і тд.

Для взаємодії з всіма репозиторіями використовується клас UnitOfWork. На рисунку 5.7 зображено приклад реалізації даного класу.

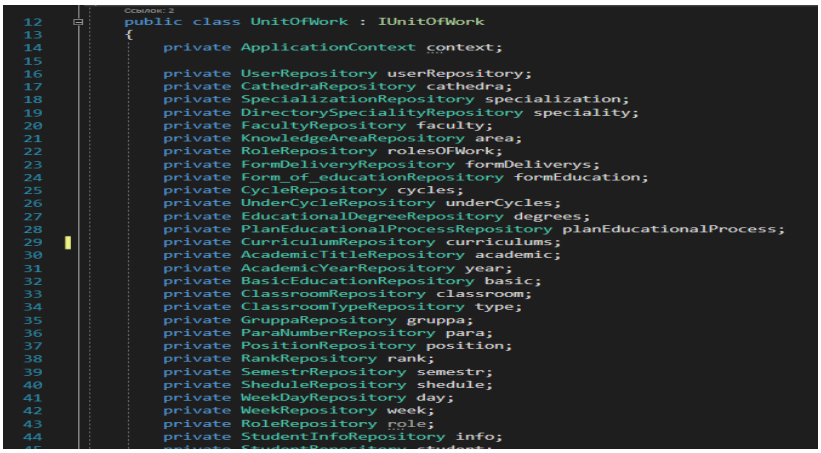

Рисунок 5.7 – Приклад класу UnitOfWork

Для створення самої бази даних по заданому сервері, потрібно використати: “Консоль диспечера пакетів”.

Спочатку потрібно прописати команду: “Add-Migration nameMigration”. Вона створить першу міграцію базі даних.

Модель даних в процесі розробки може змінитися і перестане відповідати базі даних. Ви завжди можете видалити базу даних, і EF створить для вас нову версію, в точності відповідну моделі, але така процедура призводить до втрати поточних даних. Функція міграції[15] в EF Core дозволяє послідовно застосовувати зміни схеми до бази даних, щоб синхронізувати її з моделлю даних в додатку без втрати існуючих даних. На рисунку 5.8 зображено автоматично створений клас міграції і його реалізацію.

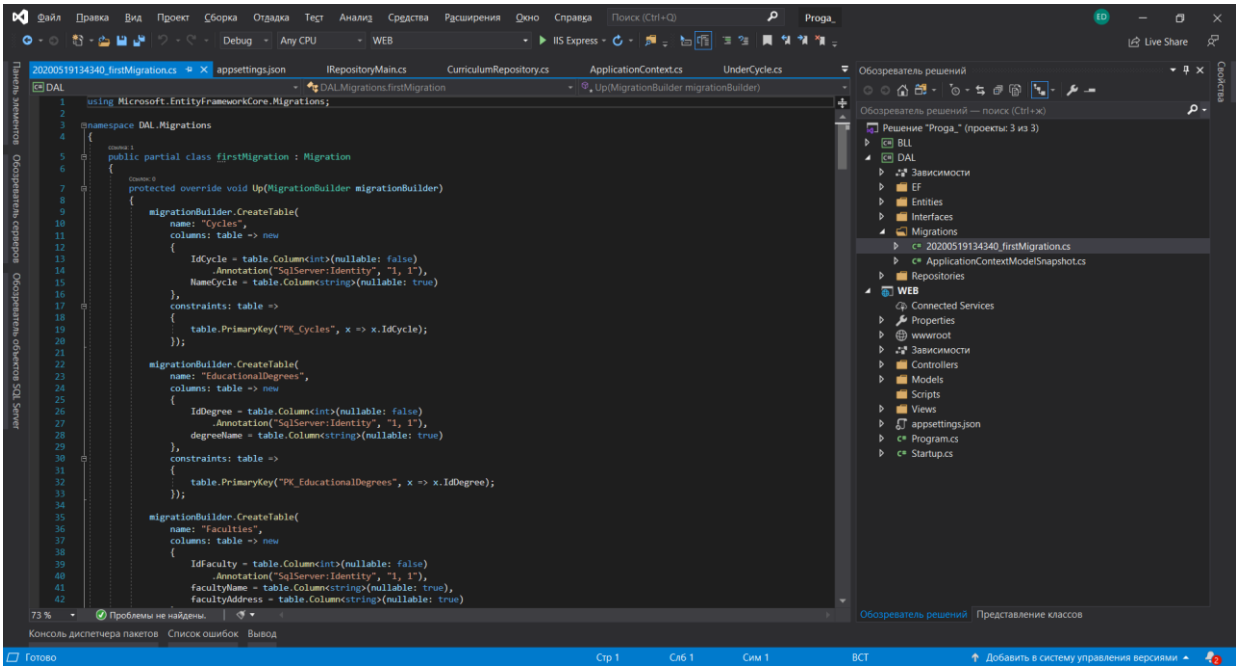

Рисунок 5.8 – Приклад класу міграції

Після цього потрібно оновити базу даних за допомогою команди: “Update-Database”. У логах виконання команди можна побачити, що з наших вище вказаних класів-моделей, створюється і виконується код мовою Sql.

На рисунку 5.9 зображено інформацію про успішне виконання команди Update-Database.

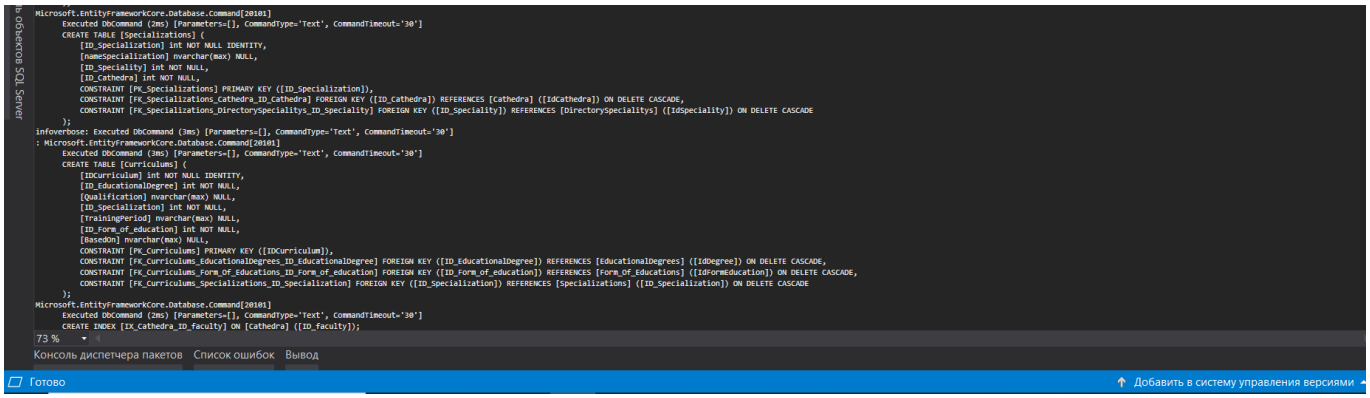

Рисунок 5.9 – Вивід про створення таблиці Curriculum.

Після виконання цих дій, база даних буде створена на сервері і можна вже з нею працювати. Оскільки представлення напряму неможе співпрацювати з моделлю бази даних, то ми перейдемо до реалізації другого рівня MVC – Business Logic Layer.

На рисунку 5.10 зображено діаграму класів для моделі даних.

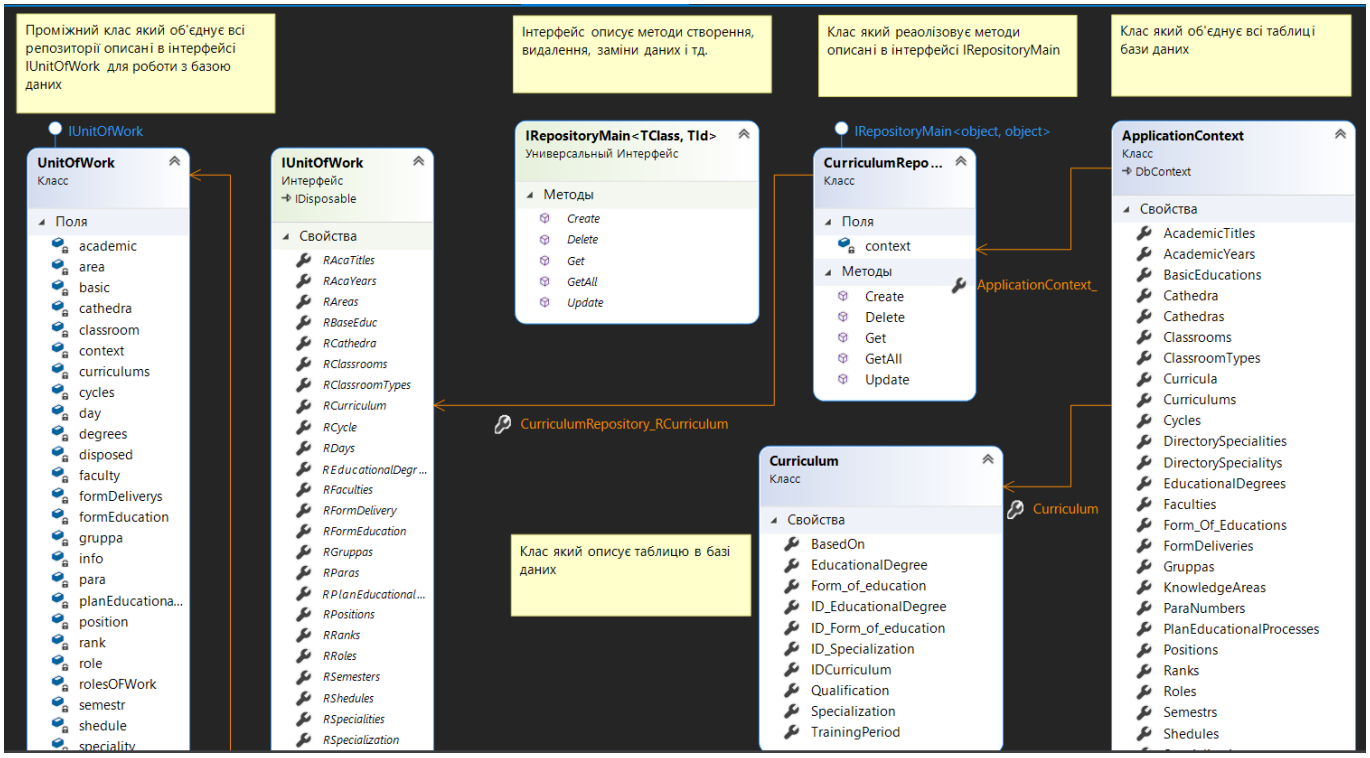

Рисунок 5.10 – Діаграма класів моделі даних

Діаграма показує всі зв'язки класу Curriculum. Зображено головні зв'язки між класами оранжевим кольором, а синім пише над класами, які інтерфейси вони наслідують. По даному прикладу будуть з'єднуватись всі інші класи в моделі даних.

5.2 Створення бізнес логіки

Business Logic Layer або бізнес-рівень інкапсулює всю бізнес-логіку, всі необхідні обчислення, отримує об'єкти з рівня доступу до даних і передає їх на рівень представлення, або, навпаки, отримує дані з рівня уявлення і передає їх на рівень даних.

Отже, додамо в рішення новий проект по типу Class Library, який назвемо NLayerApp.BLL. Оскільки бізнес-рівень буде використовувати класи з рівня доступу до даних, то нам треба додати на нього посилання. На рисунку 5.11 зображено підключення рівня моделі до рівня бізнес логіки.

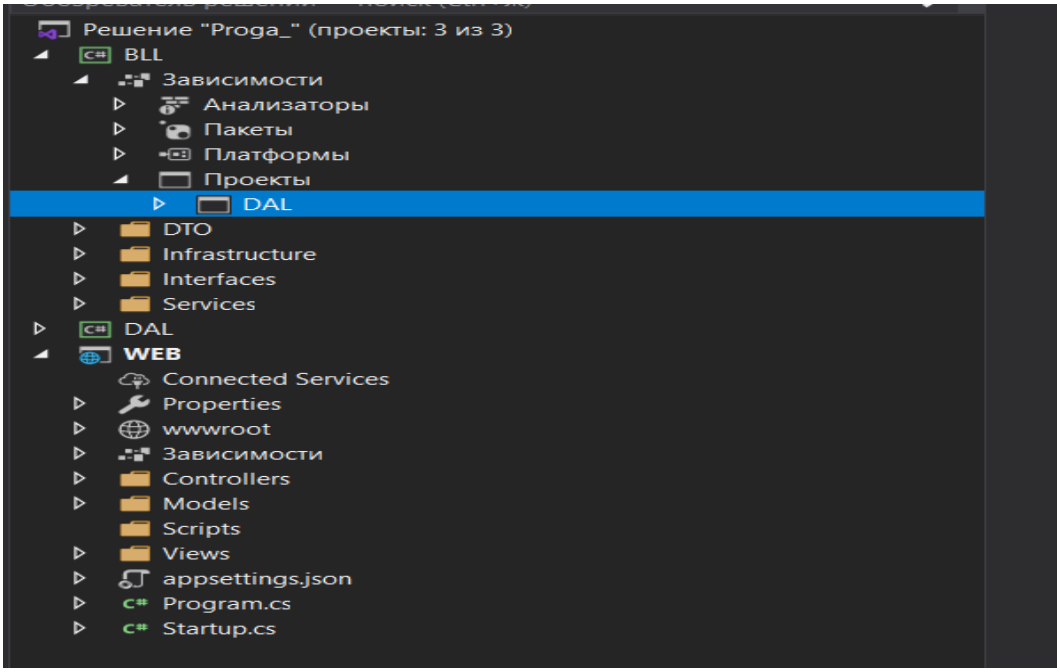

Рисунок 5.11 – Підключення DAL до BLL

Рівень представлення не може безпосередньо отримувати дані з бази даних. В даному випадку BLL виступатиме в ролі посередника між двома рівнями. Але також треба враховувати, що безпосередньо він не може передавати в контролери об'єкти Curriculum і інші, так як рівень представлення не повинен мати доступ до функціональності рівня DAL. Тому нам потрібні проміжні сутності.

Отже, додамо в проект BLL папку, яку назвемо DTO (Data Transfer Object). На риснку 5.12 зображено CurriculumDTO.

```
public class CurriculumDTO
{
    public int Id_Curriculum { get; set; }
    public string NameEducationalDegree { get; set; }
    public string NameFaculty { get; set; }
    public string NameCathedra { get; set; }
    public string NameSpesialty { get; set; }
    public string NameKnowledgeArea { get; set; }
    public string NameSpecialization { get; set; }
    public string TrainingPeriod { get; set; }
    public string BasedOn { get; set; }
    public string Qualification { get; set; }
    public string NameForm_of_education { get; set; }
}
```

Рисунок 5.12 – код класу CurriculumDTO

Через цей клас ми будемо передавати об'єкти навчальних планів між рівнями. Але хоча даний клас багато в чому схожий з визначення на клас Curriculum, це необов'язкова умова. Клас CurriculumDTO повинен містити тільки ті дані, які ми збираємося передати на рівень представлення або, навпаки, отримати з цього рівня. Тобто це те, що називається Data Transfer Object – спеціальна модель для передачі даних.

Взаємодіяти між іншими двома рівнями ми будемо через спеціальний сервіс. Знову ж для більшої гнучкості спочатку визначимо його інтерфейс. Для зберігання інтерфейсу додамо в BLL папку Interfaces і в неї покладемо інтерфейс ICurriculumService. На рисунку 5.13 зображено реалізацію інтерфейсу ICurriculumService.

```
public interface ICurriculumService : IDisposable
{
    Ссылок: 2
    CurriculumDTO FindCurriculum(string Name);
    Ссылок: 3
    IEnumerable<CurriculumDTO> GetAll();

    Ссылок: 2
    Task<OperationDetails> Create(CurriculumDTO item);
    ссылка: 1
    Task<OperationDetails> Remove(int Id);
    ссылка: 1
    Task<OperationDetails> Update(int id);
    Ссылок: 26
    new void Dispose();
}
```

Рисунок 5.13 – Реалізація інтерфейсу ICurriculumService.

Для зберігання реалізацій інтерфейсу визначимо в проекті ще одну папку Services. Додамо в неї клас сервісу CurriculumService. На рисунку 5.14 зображено реалізацію CurriculumService.

```
Ссылок: 2
public class CurriculumService : ICurriculumService
{
    Ссылок: 8
    private IUnitOfWork Database { get; set; }

    Ссылок: 0
    public CurriculumService(IUnitOfWork uow)
    {
        Database = uow;
    }

    Ссылок: 3
    public IEnumerable<CurriculumDTO> GetAll()
    {
        List<Curriculum> newlist = Database.RCurriculum.GetAll().ToList();
        List<CurriculumDTO> newlistdto = new List<CurriculumDTO>();

        for (int i = 0; i < newlist.Count; ++i)
        {
            newlistdto.Add(new CurriculumDTO
            {
                Id_Curriculum = newlist[i].IDCurriculum,
                BasedOn = newlist[i].BasedOn,
                Qualification = newlist[i].Qualification,
                TrainingPeriod = newlist[i].TrainingPeriod,
                NameCathedra = newlist[i].Specialization.Cathedra.cathedraName,
                NameEducationalDegree = newlist[i].EducationalDegree.degreeName,
                NameFaculty = newlist[i].Specialization.Cathedra.Faculty.facultyName,
                NameForm_of_education = newlist[i].Form_of_education.Name,
                NameSpecialization = newlist[i].Specialization.nameSpecialization,
                NameSpesialty = newlist[i].Specialization.Speciality.specialityName,
                NameKnowledgeArea = newlist[i].Specialization.Speciality.KnowledgeArea.areaName
            });
        }

        return newlistdto;
    }

    Ссылок: 2
    public CurriculumDTO FindCurriculum(string name)
    {
        Curriculum item = Database.RCurriculum.Get(name);
```

Рисунок 5.14 – Реалізація сервісу CurriculumService.

CurriculumService в конструкторі приймає об'єкт IUnitOfWork, через який йде взаємодія з рівнем DAL. Метод GetAll() отримує всі навчальні плани і передає на рівень представлення.

Впровадження залежностей – це патерн, який використовується для дозволу залежностей, в даному паттерні класи або об'єкти мають свої залежні класи введені (передані іншим класом або об'єктом), а не створені безпосередньо. Використовується для того, щоб максимально відокремити об'єкти і їх залежності. На рисунку 5.15 зображено реалізацію встановлення зв’язків між інтерфейсами і їхніми сервісами.

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddTransient<IUserService, UserService>();
    services.AddTransient<IUnitOfWork, UnitOfWork>();
    services.AddTransient<IFacultyService, FacultyService>();
    services.AddTransient<IKnowledgeAreaService, KnowledgeAreaService>();
    services.AddTransient<ISpecialityService, SpecialityService>();
    services.AddTransient<ICathedraService, CathedraService>();
    services.AddTransient<ISpecializationService, SpecializationService>();
    services.AddTransient<ICycleService, CycleService>();
    services.AddTransient<IUnderCycleService, UnderCycleService>();
    services.AddTransient<IFormDeliveryService, FormDeliveryService>();
    services.AddTransient<IFormEducationService, FormEducationService>();
    services.AddTransient<IEducationalDegreeService, EducationalDegreeService>();
    services.AddTransient<ICurriculumService, CurriculumService>();
    services.AddTransient<IPlanEducationalProcessService, PlanEducationalProcessService>();
```

Рисунок 5.15 – Встановлення зв’язків

Оскільки в даному випадку ми не оголошуємо в конструкторі явно об'єкт IUnitOfWork, то нам треба використовувати впровадження залежностей для передачі конкретної реалізації даного інтерфейсу в CurriculumService. То для цього нам потрібно вказати залежності явним спомобом у файлі Startup.

На рисунку 5.16 зображено діаграму класів бізнес логіки.

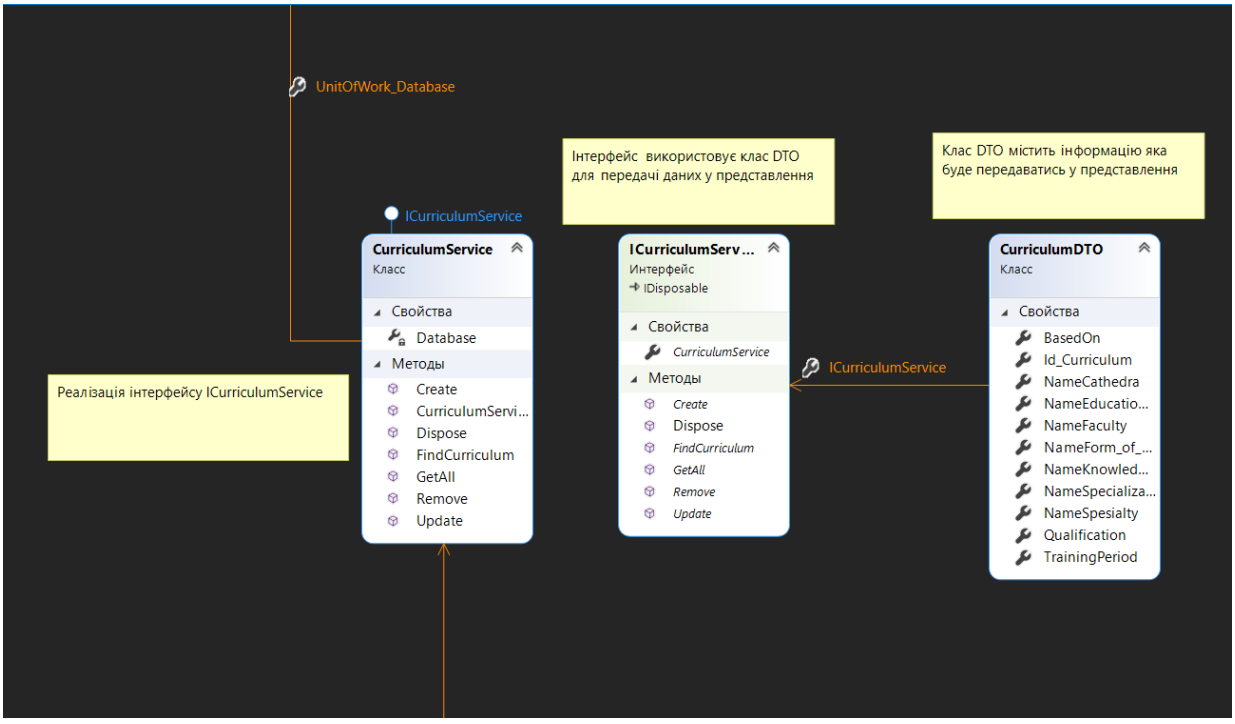

Рисунок 5.16 – Діаграма класів бізнес логіки

Зв’язок UnitOfWork_Database з’єднує бізнес логіку з попереднім рівнем – моделі бази даних. Рівень бізнес логіки виступає проміжним рівнем між моделлю і представленням.

5.3 Створення представлення

Presentation Layer або рівень представлення відповідає за взаємодію з користувачем. Для цього в нашому рішенні вже є проект WEB. Правда, фактично всю основну роботу і весь основний функціонал ми створили, залишилося тільки його застосувати на рівні представлення. Для цього спочатку додамо в проект посилання на проект BLL. Для представлення даних в представленнях визначимо моделі представлення в папці Models. На рисунку 5.17 зображено приклад моделі для навчальних планів.

```
Ссылок: 6
public class Add_CurriculumModel
{
    Ссылок: 2
    public string nameEducationalDegree { get; set; }
    Ссылок: 2
    public string Qualification { get; set; }

    Ссылок: 2
    public string nameSpecialization { get; set; }
    Ссылок: 2
    public string TrainingPeriod { get; set; }

    Ссылок: 2
    public string nameForm_of_education { get; set; }
    Ссылок: 2
    public string BasedOn { get; set; }

    Ссылок: 2
    public List<SelectListItem> newListSpecialization { get; set; }
    Ссылок: 2
    public List<SelectListItem> newListEducationalDegree { get; set; }
    Ссылок: 2
    public List<SelectListItem> newListForm_of_education { get; set; }
}
```

Рисунок 5.17 – Реалізація моделі навчальних планів

Реалізуємо контроллери. Контролер отримує введення користувача, обробляє його і посилає назад результат обробки, наприклад, у вигляді подання.

При використанні контролерів існують деякі умовності. Так, за угодами про іменування назви контролерів повинні закінчуватися на суфікс “Controller”, інша ж частина до цього суфікса вважається ім'ям контролера.

Щоб звернутися контролера з веб-браузера, нам треба в адресному рядку набрати адреса_сайту / Ім’я_контроллера /. На рисунку 5.18 зображено реалізацію CurriculumController.

```
public class CurriculumController : Controller
{
    private readonly ICurriculumService _Service;
    private readonly ISpecializationService _ServiceS;
    private readonly IFormEducationService _ServiceF;
    private readonly IEducationalDegreeService _ServiceE;

    public CurriculumController(ICurriculumService serv, ISpecializationService _S, IFormEducationService _F, IEducationalDegreeService _E)
    {
        _Service = serv;
        _ServiceS = _S;
        _ServiceE = _E;
        _ServiceF = _F;
    }

    public IActionResult GetCurriculum()
    {
        CurriculumsModel newModel = new CurriculumsModel();
        newModel.newList = _Service.GetAll().ToList();
        return View("~/Views/Curriculum/Curriculums.cshtml", newModel);
    }

    public async Task<IActionResult> CreateNewCurriculum(Add_CurriculumModel model)
    {
        CurriculumDTO newItem = new CurriculumDTO()
        {
            Qualification = model.Qualification,
            NameSpecialization = model.nameSpecialization,
            NameEducationalDegree = model.nameEducationalDegree,
            NameForm_of_education = model.nameForm_of_education,
            TrainingPeriod = model.TrainingPeriod,
            BasedOn = model.BasedOn
        };

        await _Service.Create(newItem);
        return RedirectToAction("GetCurriculum", "Curriculum");
    }

    public IActionResult Add_Curriculum()
```

Рисунок 5.18 – Приклад контроллера який відповідає за навчальні плани

Даний контроллер має методи для відображення всіх навчальних планів, створення нового. Ми в контроллері працюємо з моделлю даних, яка передається в представлення або отримується з нього. На рисунку 5.19 зображено реалізацію представлення всіх навчальних планів.

```
<table>
    <tr>
        <th colspan="11" class="text1">Міністерство освіти і науки</th>
    </tr>
    <tr>
        <th colspan="11" class="text2">НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ "КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО</
    </tr>
    <tr>
        <th colspan="11" class="text3">НАВЧАЛЬНИЙ ПЛАН</th>
    </tr>

    <tr>
        <th>ID</th>
        <th>Підготовка</th>
        <th>Галузь знань</th>
        <th>Факультет</th>
        <th>Спеціальність</th>
        <th>Кваліфікація</th>
        <th>Спеціалізація</th>
        <th>Термін навчання</th>
        <th>Форма навчання</th>
        <th>На основі</th>
        <th>Кафедра</th>
    </tr>
    <tr>
        <td>@Model.curriculum.Id_Curriculum </td>
        <td>@Model.curriculum.NameEducationalDegree </td>
        <td>@Model.curriculum.NameKnowledgeArea</td>
        <td>@Model.curriculum.NameFaculty</td>
        <td>@Model.curriculum.NameSpesialty </td>
        <td>@Model.curriculum.Qualification </td>
        <td>@Model.curriculum.NameSpecialization</td>
        <td>@Model.curriculum.TrainingPeriod</td>
        <td>@Model.curriculum.NameForm_of_education</td>
        <td>@Model.curriculum.BasedOn</td>
        <td>@Model.curriculum.NameCathedra</td>
    </tr>
</table>

<table>
    <tr>
        <th colspan="14"> План освітнього процесу</th>
    </tr>
```

Рисунок 5.19 Приклад реалізації представлення.

Дане представлення приймає модель даних з контроллера і виводить дані у формі таблиці.

Таким чином, ми створили програму, що реалізує трирівневу архітектуру, яке виконує всі ті ж дії, що і проект з монолітної архітектурою. Однак тепер нам легше його тестувати, розробляти в команді розподілено, розвивати і розширювати.

Для створення динамічних списків (вибір кафедри в залежності від факультету) використаємо ajax. AJAX[16] (Асинхронний JavaScript і XML) являє собою технологію гнучкого взаємодії між клієнтом і сервером. Завдяки її використанню ми можемо здійснювати асинхронні запити до сервера без перезавантаження всієї сторінки. Правда, в даний час все більше замість формату XML використовується формат JSON для взаємодії між клієнтом і сервером.

Для підключення до проекту використаємо NuGet, завантажимо пакет jQuery.Ajax.Unobtrusive. На рисунку 5.20 зображено інформацію про пакет jQuery.Ajax.Unobtrusive.

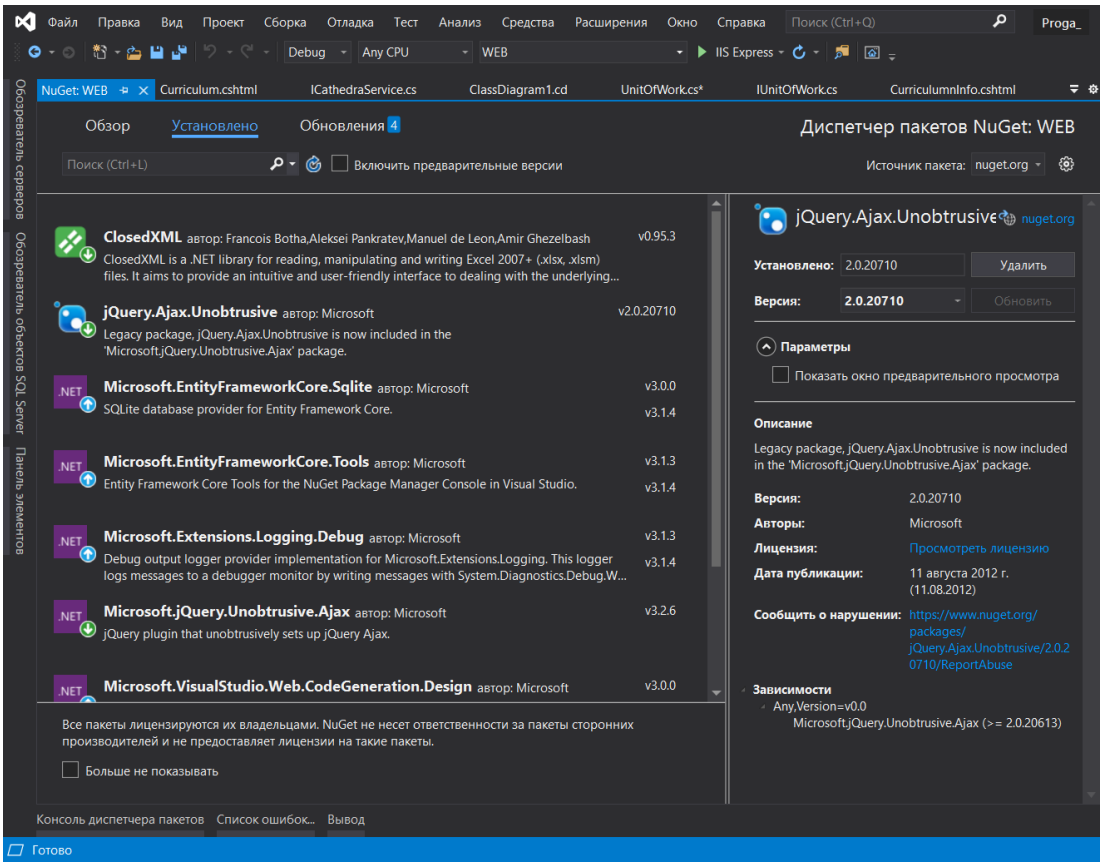

Рисунок 5.20 – Встановлення пакету jQuery.Ajax.Unobtrusive

У файлі представлення добавимо новий scripts, який буде містити обробку події зміни інформації у випадаючому списку. На рисунку 5.21 зображено реалізацію події.

```
@section scripts{
    <script type="text/javascript">
        $(function ()
        {
            $('#faculty').change(function()
            {
                var w = $(this).val();
                $.ajax({
                    type: 'GET',
                    url: '@Url.Action("GetItemsFaculty", "CurriculumDepartment")/' + w,
                    success: function (data) {
                        $('#cathedra').replaceWith(data);

                        $('#cathedra').change(function()
                        {
                            var w = $(this).val();
                            var q = $('#speciality').val();
                            var e = q + '&' + w;

                            $.ajax({
                                type: 'GET',
                                url: '@Url.Action("GetItemsSpecialization", "CurriculumDepartment")/' + e,
                                success: function (data) {
                                    $('#specialization').replaceWith(data);
                                }
                            });
                        });
```

Рисунок 5.21 – Реалізація події зміни значення у списку

При зміні інформації списку буде створюватись новий запит і оброблятися контроллером. Результатом виконання буде заміна існуючого списку з певними значеннями на новий список, для цього використовується часткове представлення. На рисунку 5.22 зображено реалізацію часткового представлення для динамічного списку.

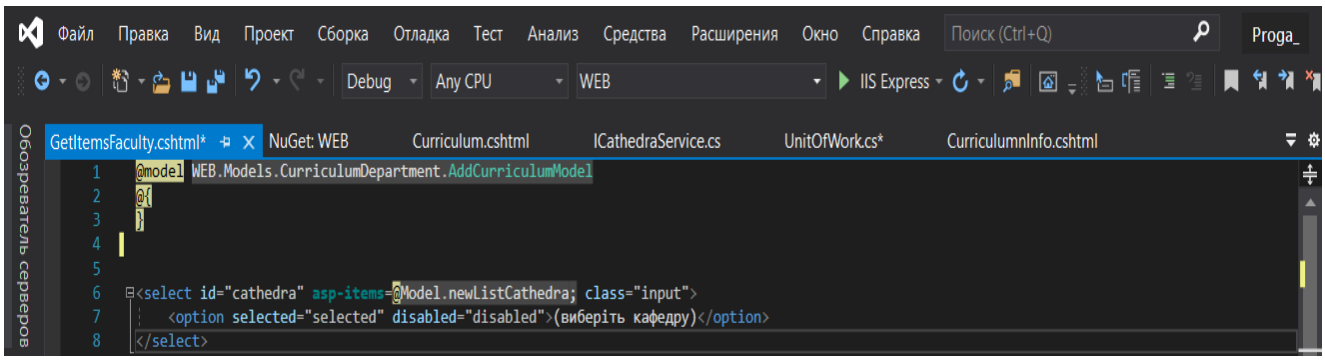

Рисунок 5.22 – Реалізація часткового представлення

На рисунку 5.23 зображено діаграму класів для представлення. Зв’язок CurriculumServise з’єднує контроллер представлення з рівнем бізнес логіки.

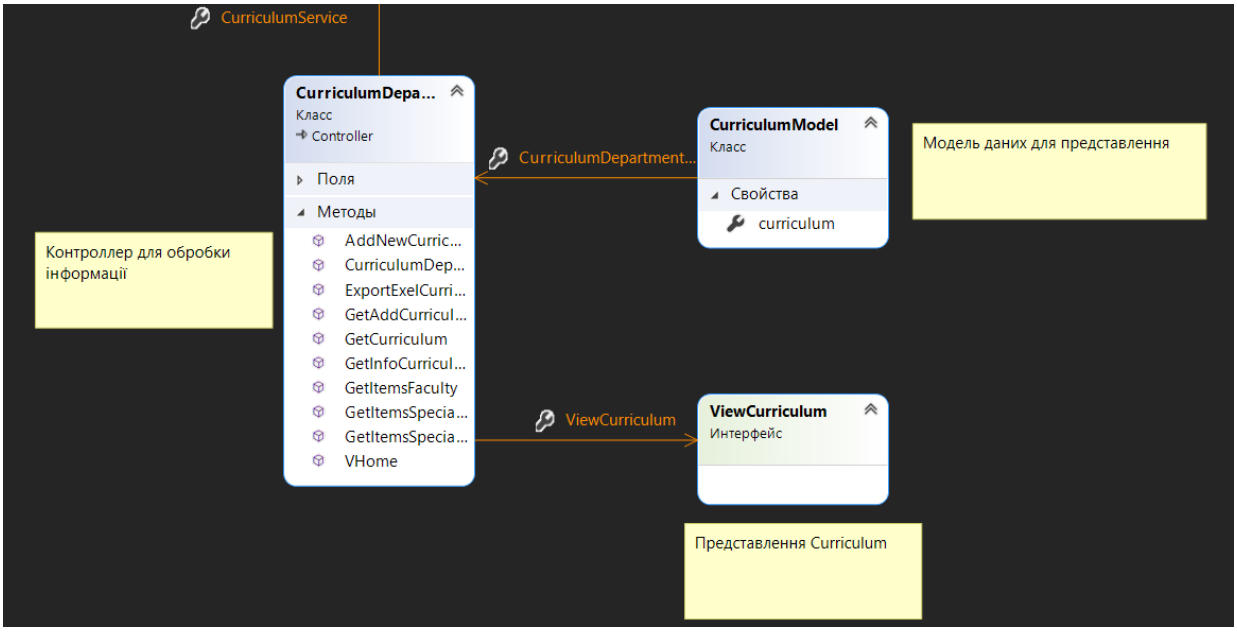

Рисунок 5.23 – Діаграма класів представлення

5.4 Кольорова гама

Вибір кольорів для використання у дизайні є однією із найважливіших частин створення дизайну користувацького інтерфейсу[17]. Від відтинку кольорів та їх поєднання між собою.

Для гарного взаємного контрасту пари кольорів, вони повинні бути обраними із протилежних частин кольорового спектру.

Основним кольором було обрано помаранчевий. Він суб’єктивно асоціюється у користувачів із бадьорістю та активністю, що буде спонукати їх більш активно користуватися сервісом. Увесь помаранчевій колір, що представлено у дизайні різними відтінками, щоб складати враження впевненості.

Як додаткові фонові кольорі використовуються чорний та білий, що складають класичну пару та доповнюють використання основного кольору, для кращої візуалізації.

5.5 Висновки

У даному розділі було переведено увагу з архітектурних рішень до рішень проектування та реалізації зовнішнього вигляду користувацької частини веб-застосунку. Було спроектовано трьохрівневу архутектуру, що дозволило реалізовувати кожен елемент користувацького інтерфейсу окремо. Таке послаблення зв’язності полегшує проектування коду та його повторне використання. Програму легко тестувати, розширювати доступний функціонал, змінювати таблиці в базі даних без великих затрат часу. За допомогою технології Entity зручно працювати з даними в базі даних.

Приділення уваги до кольорів та ергономіки повинно покращити враження користувачів від веб-застосунку та спонукати їх активно користуватися сервісом.

6 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Для забезпечення безвідмовної роботи системи треба дотримуватися основних вимог та рекомендацій щодо її використання.

6.1 Системні вимоги та інсталяція

Встановлення системи непотрібне оскільки вона створена у веб версії і доступна за посиланням.

6.2 Сценарій роботи користувача з системою

Користувач взаємодіє з графічним інтерфейсом, що доступний як веб додаток. Оскільки у системі є можливості роботи розподілені по ролях, тому початковим вікном є авторизація. На рисунку 6.1 зображено початкове вікно.

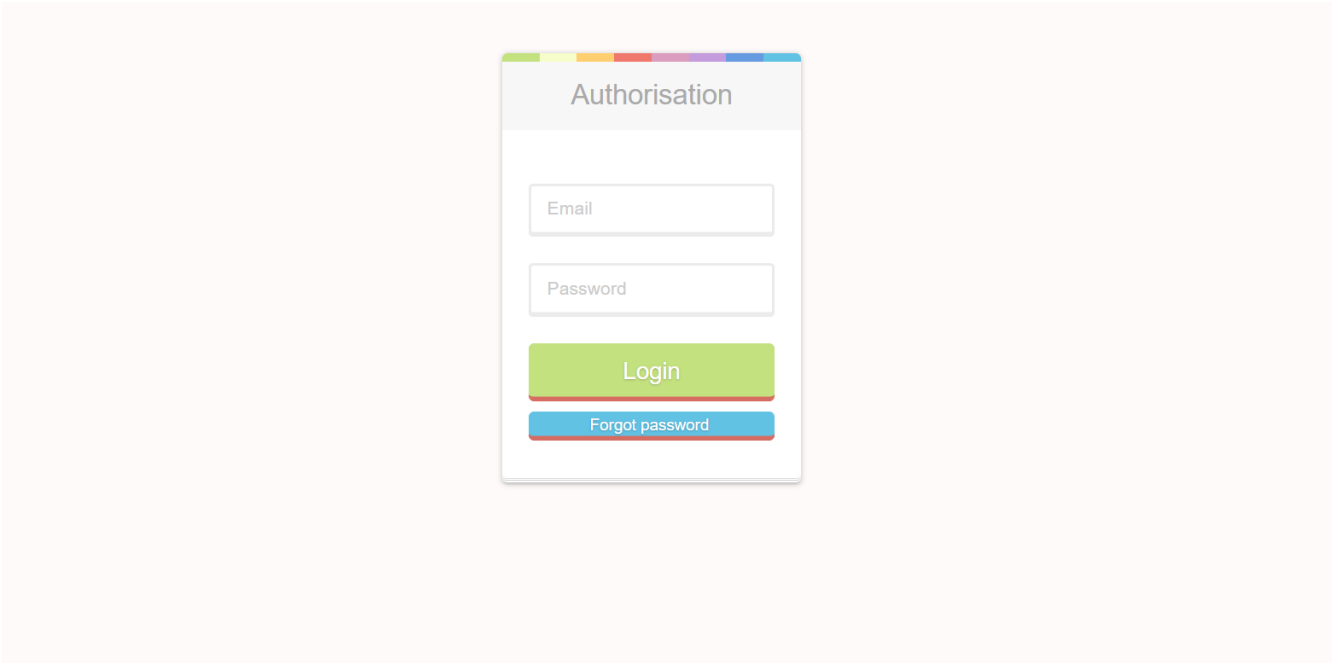

Рисунок 6.1 – Початкове вікно системи.

Авторизувавшись як керівник по створенню навчальних планів буде доступний функціонал зображений на рисунку 6.2.

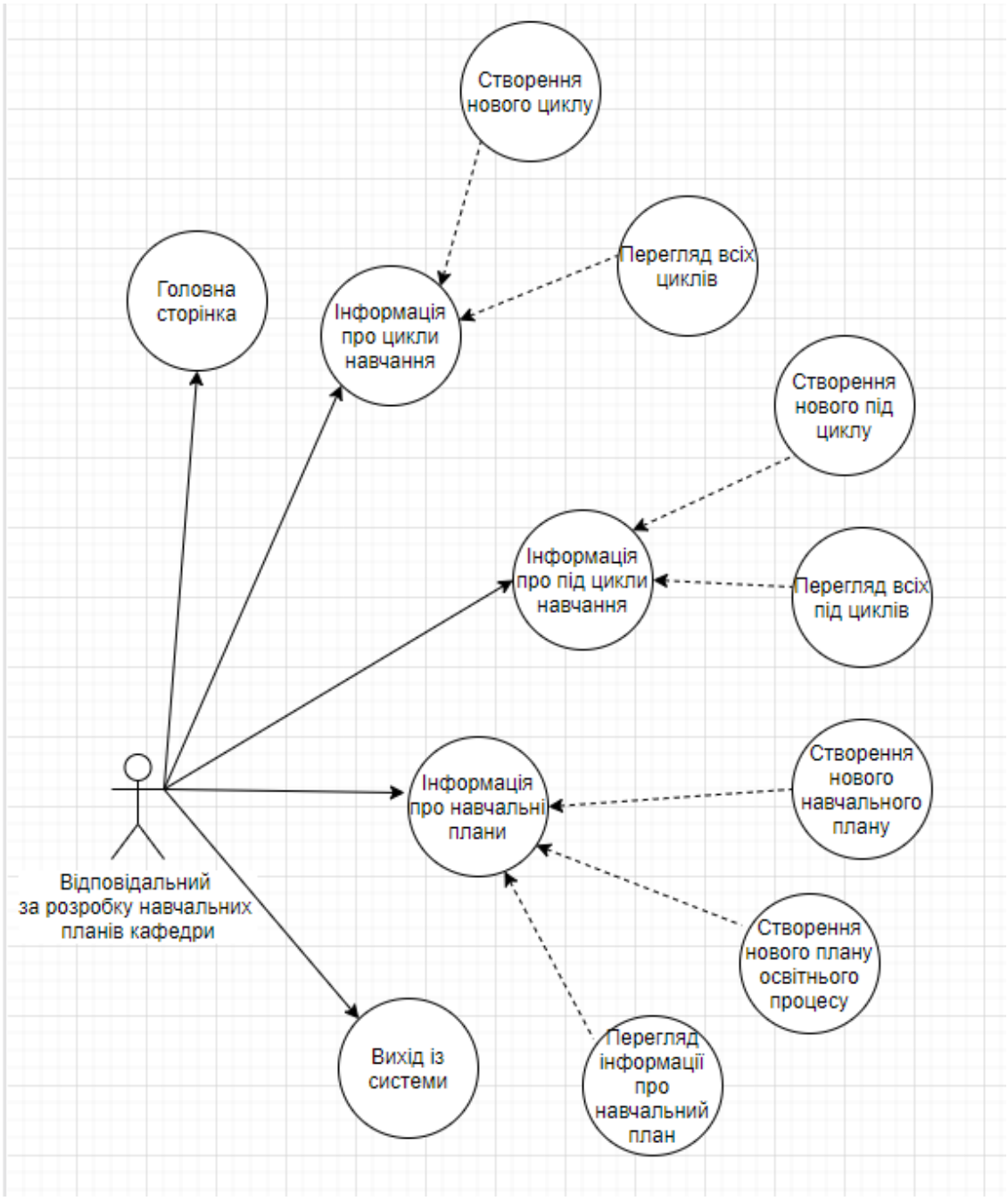

Рисунок 6.2 – Діаграма прецедентів доступного функціоналу для користувача

На головній сторінці зображено коротку інформацію про Київський політехнічний інститут імені Ігоря Сікорського. На рисунку 6.3 зображено головну сторінку.

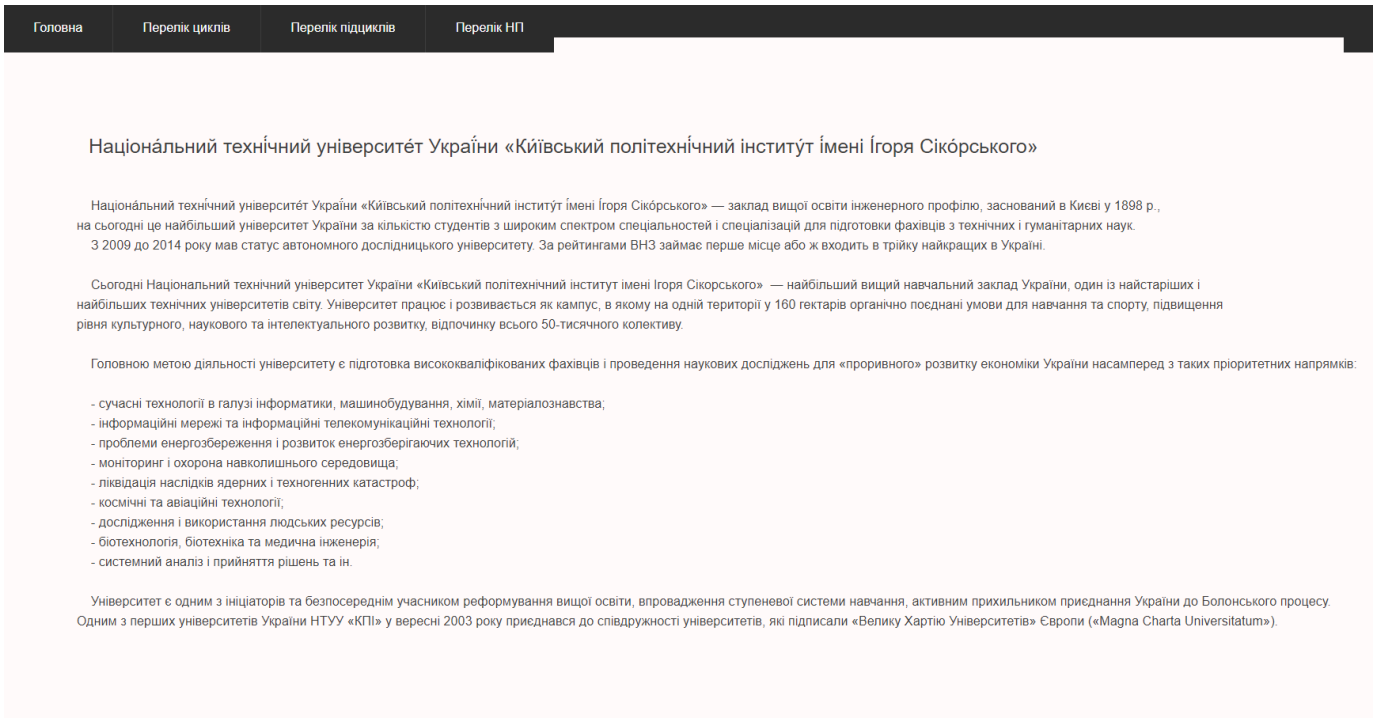

Рисунок 6.3 – Головна сторінка системи

Користувач має можливість створити новий цикл плану освітнього процесу (цикл загальної підготовки, цикл професійної підготовки). Для зручності в користуванні і для того щоб з групувати функціонал, використовується випадаюче меню. На рисунку 6.4 зображено випадаюче меню для створення нового циклу.

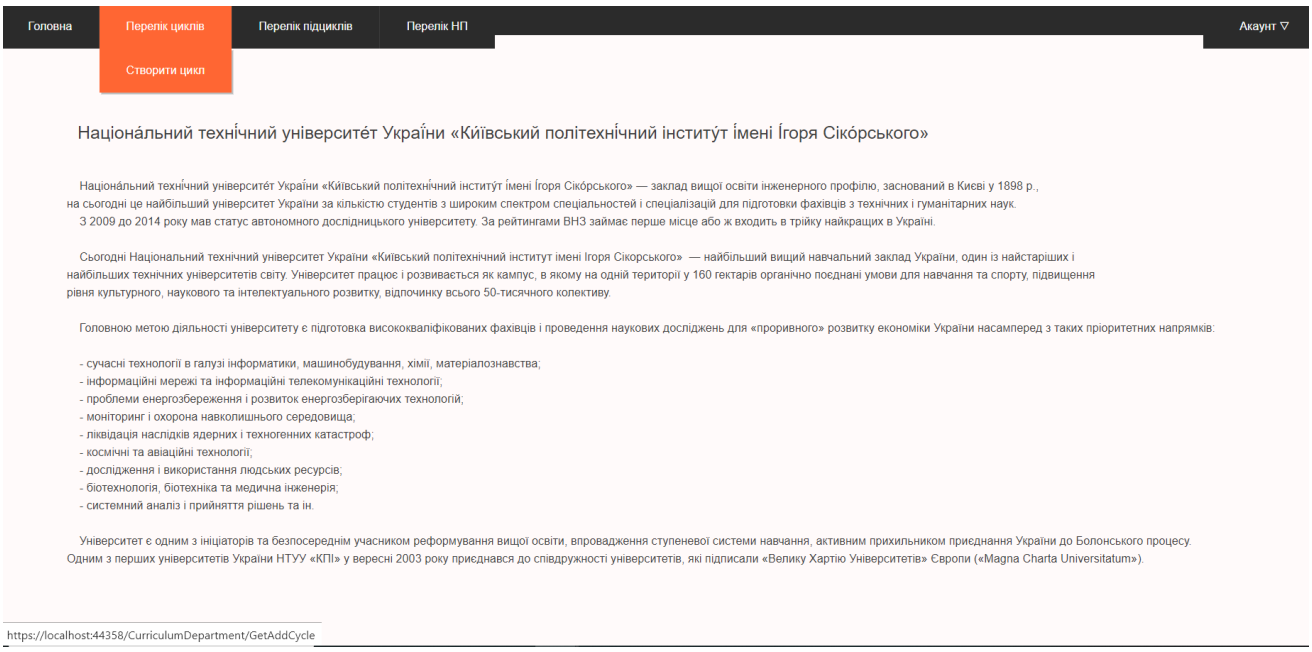

Рисунок 6.4 – Випадаюче меню для створення нового циклу

Для створення самого циклу використовується окрема сторінка, яка зображена на рисунку 6.5.

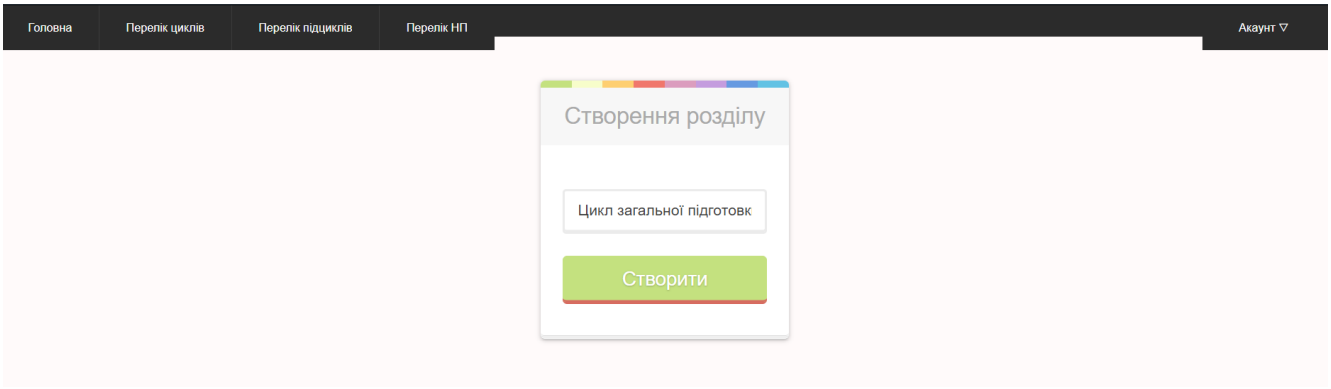

Рисунок 6.5 – Форма для створення нового циклу

Переглянути всю інформацію про цикли можна на відповідній сторінці, яка зображена на рисунку 6.6.

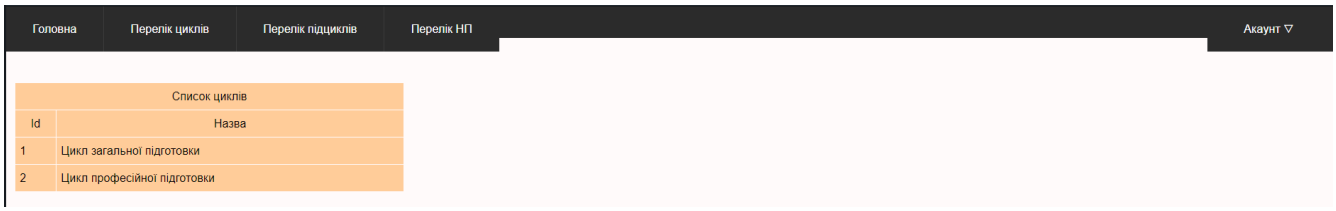

Рисунок 6.6 – Форма для виводу інформації про цикли

На рисунку 6.7 зображено можливість створення нових підциклів (навчальні дисципліни природничо-наукової підготовки, навчальні дисципліни базової підготовки).

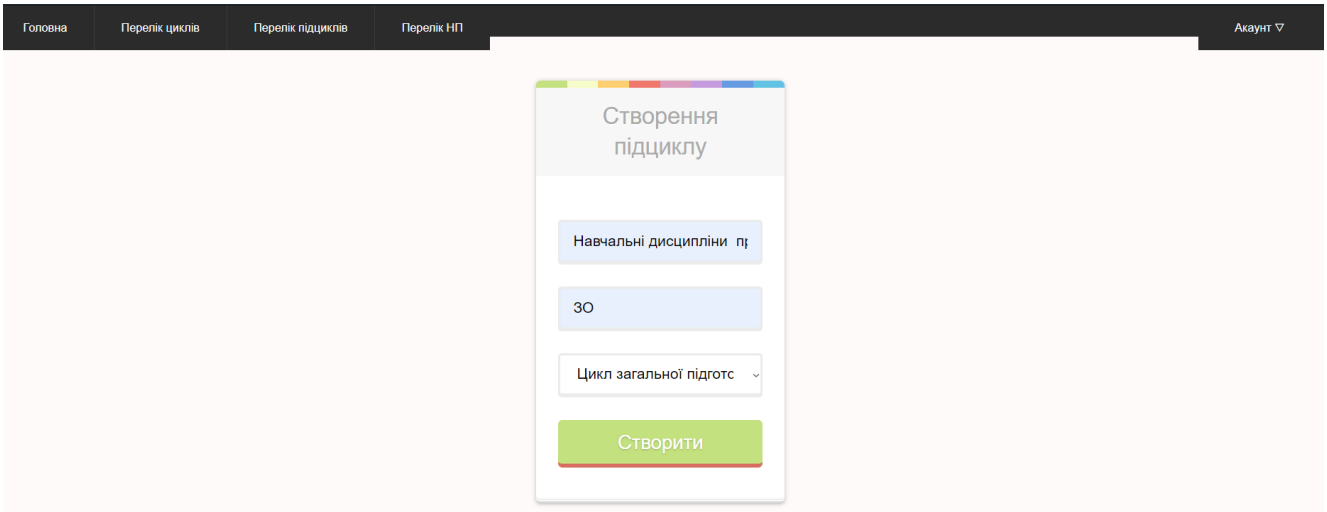

Рисунок 6.7 – Форма для створення нового підциклу

Для забезпечення зв’язку між циклом і підциклом використовується випадаюче меню, яке дозволяє вибрати цикл вже з існуючих.

На рисунку 6.8 зображено інформацію про всі підцикли.

Рисунок 6.8 – Форма для виводу інформації про підцикли

Таблиця вміщає інформацію про найменування циклу, підциклу і коду освітнього плану.

На рисунку 6.9 зображено форму для створення нового навчального плану.

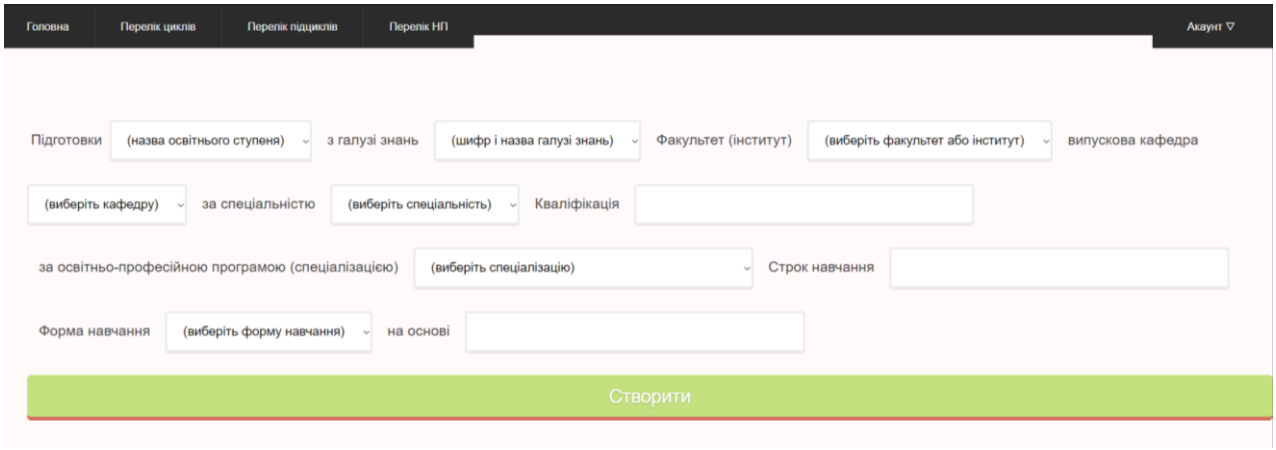

Рисунок 6.9 – Форма для створення нового навчального плану

Форма містить всю необхідну інформацію для створення нового навчального плану. Для зручного заповнення даних інформація автоматично завантажується у випадаючі списки взалежності від вибору. На рисунках 6.10-6.11 зображено залежність між вибором факультету і кафедри.

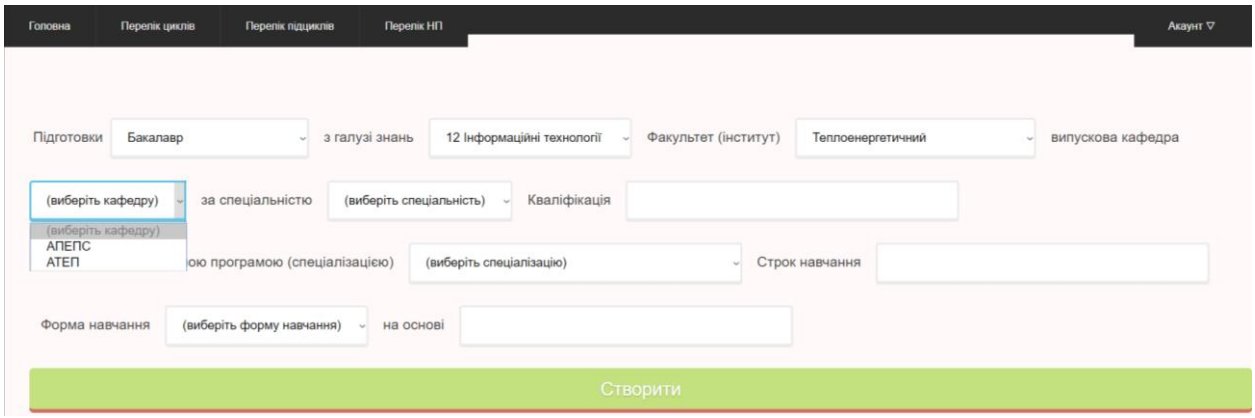

Рисунок 6.10 – Доступний вибір кафедр для теплоенергетичного факультету

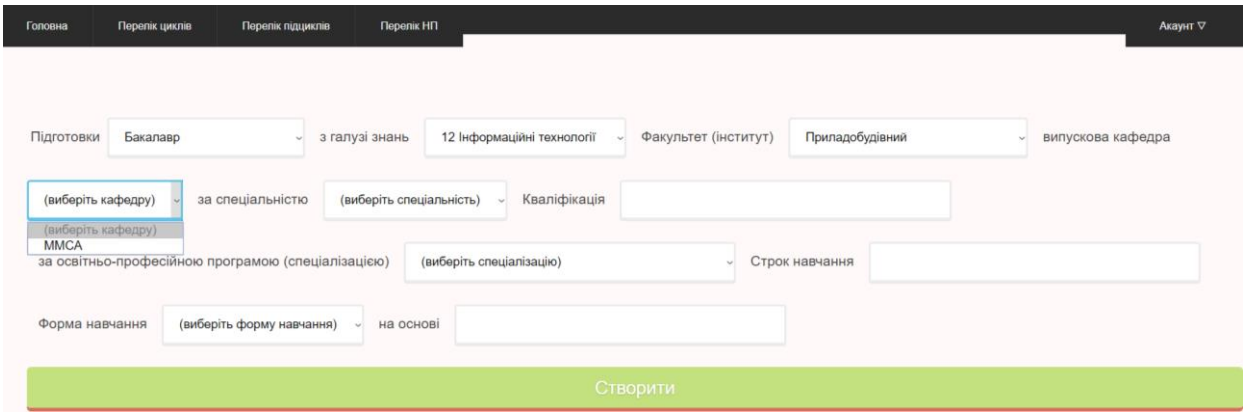

Рисунок 6.11 – Доступний вибір кафедр для приладобудівного факультету

По даній аналогії працюють всі інші зв’язані випадаючі списки.

Після створення інформації про навчальний план потрібно заповнити план освітнього процесу дисциплінами. На рисунку 6.2 зображено форму для створення дисципліни.

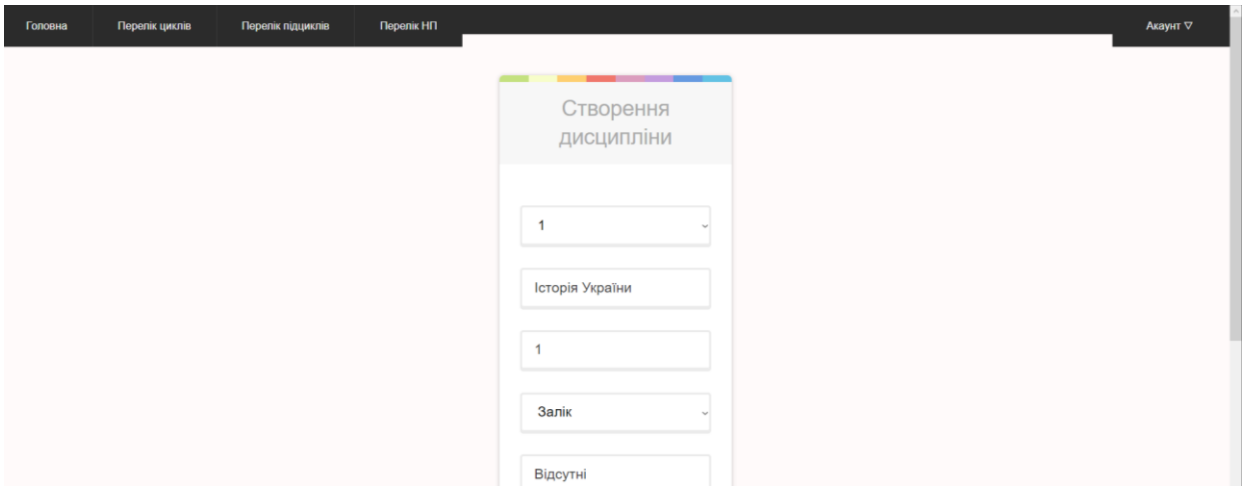

Рисунок 6.12 – Форма для створення дисциплін

Щоб переглянути інформацію про певний навчальний план, потрібно вибрати зі списку всіх навчальних планів потрібний. На рисунку 6.13 зображено список всіх навчальних планів.

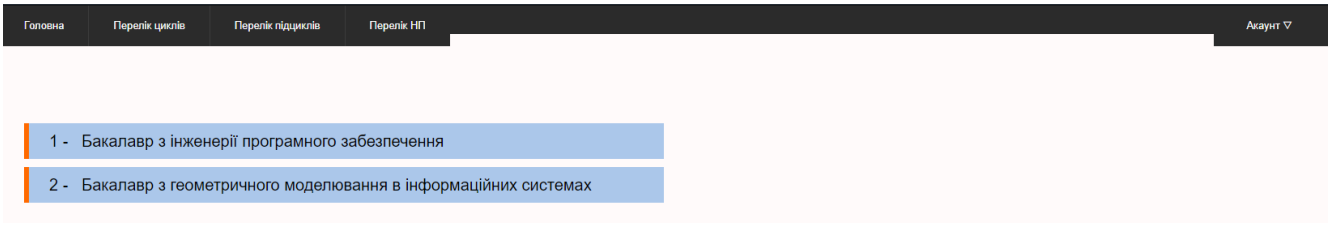

Рисунок 6.13 – Список всіх навчальних планів

Інформацію про обраний навчальний план зображено на рисунку 6.14.

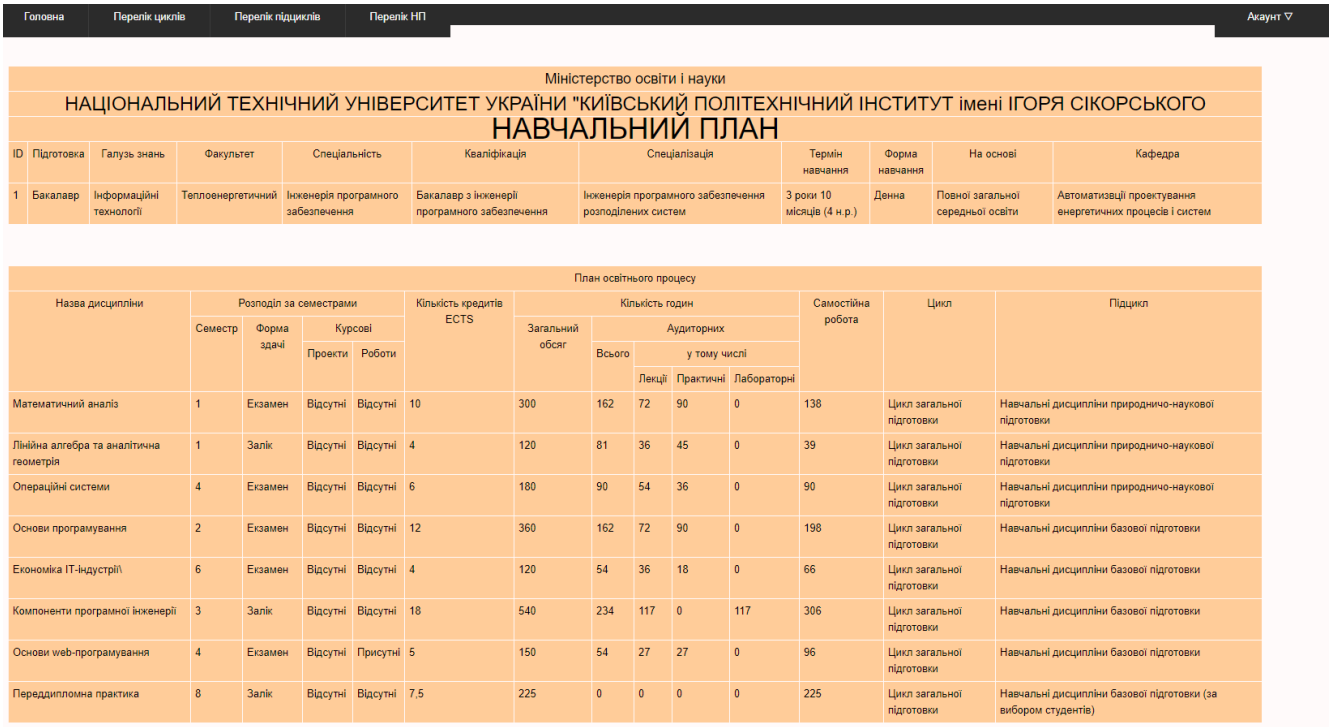

Рисунок 6.14 – Інформація про обраний навчальний план зі списку

Отже, дана система має можливість вносити всі необхідні дані для створення навчальних планів. Розроблені інтерфейси для внесення і перегляду даних. Система розбити на ролі і їх доступний функціонал. Оскільки використана трьохрівнева архітектура, проект легко розширювати.

ВИСНОВКИ

В ході виконання даної роботи було створено модуль розробки навчальних планів, як складову частину інформаційної системи кафедри.

Було досягнуто результатів:

1. Проведено аналіз процесу розробки навчальних планів на основі тимчасового положення щодо організації освітнього процесу в КПІ імені Ігоря Сікорського, що дозволило виділити інформацію, з якою буде оперувати модуль системи.
2. Виконано огляд існуючих програмних рішень для розробки навчальних планів (“Ректор 3”, “Курс: ВНЗ”), які підтвердили актуальність розробки системи.
3. Для реалізації модуля системи обрано середовище Visual Studio 2019, застосовано технології Entity Framework та MVC, бо вони забезпечують стабільну роботу на різних платформах і роблять систему зручною, для подальшої розробки.
4. Розроблено модель бази даних Ms Sql, з використанням реляційних технологій, для збереження всіх необхідних даних для роботи системи. База даних має переваги в швидкості своєї роботи та можливості зберігати велику кількість даних.
5. Створено програмний продукт, який забезпечує роботу зі створення та редагування навчальних планів. Продукт має трьохрівневу архітектуру, що дозволяє легко його модернізувати, тестувати та допрацьовувати в подальшому.
6. Сформовано опис роботи користувача з розробленим програмним продуктом. У якості користувача виступають відповідальні за розробку навчальних планів викладачі кафедри.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Рекомендації щодо розроблення навчальних планів / Уклад. В.П. Головенкін. – К.: НТУУ «КПІ», 2012. – 28 с. – 250 прим.

2. Рекомендації користувачам щодо роботи в програмі АСПНП | освітній процес в КПІ. ім. Ігоря Сікорського [Електронний ресурс] – Режим доступу до ресурсу: https://osvita.kpi.ua/node/16.

3. Расписание занятий: “Ректор-ВУЗ” [Електронний ресурс] – Режим доступу до ресурсу: http://rector.spb.ru/raspisanie-vuz-4u.php.

4. Куликов, С. С. К90 Работа с MySQL, MS SQL Server и Oracle в примерах: практ. пособие. / С. С. Куликов. – Минск: БОФФ, 2016. – 339 с.

5. IDE Visual Studio 2019 – программное обеспечение для Windows [Електронний ресурс] – Режим доступу до ресурсу: https://visualstudio.microsoft.com/ru/vs/.

6. Learning JavaScript by Ethan Brown Printed in the United States of America. Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol 2016. – 306 с.

7. Введение в ASP.NET Core [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/sharp/aspnet5/1.1.php.

8. Microsoft Corporation AMERICAN COMPANY [Електронний ресурс] – Режим доступу до ресурсу: https://www.britannica.com/topic/Microsoft-Corporation.

9. Itzik Ben-Gan, T-SQL Fundamentals, Third Edition, Printed in the United States of America 2016. – 20 с.

10. ASP.NET MVC Pattern [Електронний ресурс] – Режим доступу до ресурсу: https://dotnet.microsoft.com/apps/aspnet/mvc.

11. Smith Jon. Entity Framework Core in Action, Manning Publications, 2018. – 58 с.

12. Mapping Objects to Relational Databases [Електронний ресурс] – Режим доступу до ресурсу: http://www.agiledata.org/essays/mappingObjects.html.

13. LINQ: язык интегрированных запросов в C# 2010 для профессионалов. : Пер. англ. – М. : ООО “И.Д. Вильямс”, 2011. – 550 с.

14. Julia Lerman and Rowan Miller, Programming Entity Framework: Printed in the United States of America 2012. – 171 с.

15. Миграции [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/ru-ru/ef/core/managing-schemas/migrations.

16. Овчаренко А. В. О-35 Ajax на примерах. – СПб.: БХВ-Петербург, 2009. – 398 с.

17. Терри Фельке-Моррис, Большая книга веб-дизайна, М. Єксемо, 2012 – 218 с.

18. Матеріали VIII Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених з автоматичного управління, присвячено Дню космонавтики. Видавництво ФОП Вишемирський В.С., Херсон 2020. – 92 с.

ДОДАТОК А

Модуль розробки навчальних планів інформаційної системи автоматизованої підтримки навчальної діяльності кафедри

Специфікація

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61106_20Б

Аркушів 2

Київ – 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61106_20Б 81-1	Пояснювальна записка.docx	Пояснювальна записка
Комплекс		
Компоненти		
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61106_20Б 13-1	Додаток В	Опис програмного коду
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61106_20Б 12-1	Projekt.sln	Основний компонент, що включає в себе інші

ДОДАТОК Б

Модуль розробки навчальних планів інформаційної системи автоматизованої підтримки навчальної діяльності кафедри

Лістинг програми

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61106_20Б

Аркушів 10

Київ – 2020

```
public partial class Curriculum
{
        [Key]
        public int IDCurriculum { get; set; }
        [ForeignKey("EducationalDegree")]
        public int ID_EducationalDegree { get; set; }
        public string Qualification { get; set; }
        [ForeignKey("Specialization")]
        public int ID_Specialization { get; set; }
        public string TrainingPeriod { get; set; }
        [ForeignKey("Form_of_education")]
        public int ID_Form_of_education { get; set; }
        public string BasedOn { get; set; }
        public EducationalDegree EducationalDegree { get; set; }
        public Specialization Specialization { get; set; }
        public Form_of_education Form_of_education { get; set; }
}

 public class CurriculumRepository : IRepositoryMain<Curriculum, string>
{
        private ApplicationContext context;
        public CurriculumRepository(ApplicationContext applicationContext) => this.context = applicationContext;

        public void Create(Curriculum item)
        {
           context.Curriculums.Add(item);
        }

        public void Delete(string id)
        {
           context.Curriculums.Remove(item);
        }

        public Curriculum Get(string id)
        {
           var _findItem = context.Curriculums.Where(p => p.IDCurriculum == Convert.ToInt32(id)).SingleOrDefault();
           if (_findItem != null)
           {
              _findItem.Specialization = context.Specializations.Where(p => p.ID_Specialization ==
                      _findItem.ID_Specialization).SingleOrDefault();
              _findItem.Specialization.DirectorySpeciality = context.DirectorySpecialitys.Where(p => p.IdSpeciality ==
                      _findItem.Specialization.ID_Speciality).SingleOrDefault();
```

```
        _findItem.Specialization.Cathedra = context.Cathedra.Where(p => p.IdCathedra ==
                _findItem.Specialization.ID_Cathedra).SingleOrDefault();
        _findItem.Specialization.DirectorySpeciality.KnowledgeArea = context.KnowledgeAreas.Where(p => p.IdArea ==
                _findItem.Specialization.DirectorySpeciality.ID_KnowledgeArea).SingleOrDefault();
        _findItem.Specialization.Cathedra.Faculty = context.Faculties.Where(p => p.IdFaculty ==
                _findItem.Specialization.Cathedra.ID_faculty).SingleOrDefault();
        _findItem.EducationalDegree = context.EducationalDegrees.Where(p => p.IdDegree ==
                _findItem.ID_EducationalDegree).SingleOrDefault();
        _findItem.Form_of_education = context.Form_Of_Educations.Where(p => p.IdFormEducation ==
                _findItem.ID_Form_of_education).SingleOrDefault();
    }
    return _findItem;
}

public IEnumerable<Curriculum> GetAll()
{
    List<Curriculum> newList = context.Curriculums.ToList();

    for (int i = 0; i < newList.Count; ++i)
    {
        newList[i].Specialization = context.Specializations.Where(p => p.ID_Specialization ==
                newList[i].ID_Specialization).SingleOrDefault();
        newList[i].Specialization.DirectorySpeciality = context.DirectorySpecialitys.Where(p => p.IdSpeciality ==
                newList[i].Specialization.ID_Speciality).SingleOrDefault();
        newList[i].Specialization.Cathedra = context.Cathedra.Where(p => p.IdCathedra ==
                newList[i].Specialization.ID_Cathedra).SingleOrDefault();
        newList[i].Specialization.DirectorySpeciality.KnowledgeArea = context.KnowledgeAreas.Where(p => p.IdArea ==
                newList[i].Specialization.DirectorySpeciality.ID_KnowledgeArea).SingleOrDefault();
        newList[i].Specialization.Cathedra.Faculty = context.Faculties.Where(p => p.IdFaculty ==
                newList[i].Specialization.Cathedra.ID_faculty).SingleOrDefault();
        newList[i].EducationalDegree = context.EducationalDegrees.Where(p => p.IdDegree ==
                newList[i].ID_EducationalDegree).SingleOrDefault();
        newList[i].Form_of_education = context.Form_Of_Educations.Where(p => p.IdFormEducation ==
                newList[i].ID_Form_of_education).SingleOrDefault();
    }
    return newList;
}

public void Update(Curriculum item)
{
    context.Curriculums.Update(item);
}
```

```
}

public class CurriculumDTO
{
    public int Id_Curriculum { get; set; }
    public string NameEducationalDegree { get; set; }
    public string NameFaculty { get; set; }
    public string NameCathedra { get; set; }
    public string NameSpesialty { get; set; }
    public string NameKnowledgeArea { get; set; }
    public string NameSpecialization { get; set; }
    public string TrainingPeriod { get; set; }
    public string BasedOn { get; set; }
    public string Qualification { get; set; }
    public string NameForm_of_education { get; set; }
}

public class CurriculumService : ICurriculumService
{
    private IUnitOfWork Database { get; set; }

    public CurriculumService(IUnitOfWork uow)
    {
      Database = uow;
    }

    public IEnumerable<CurriculumDTO> GetAll()
    {
      List<Curriculum> newlist = Database.RCurriculum.GetAll().ToList();
      List<CurriculumDTO> newlistdto = new List<CurriculumDTO>();

      for (int i = 0; i < newlist.Count; ++i)
      {
        newlistdto.Add(new CurriculumDTO
        {
          Id_Curriculum = newlist[i].IDCurriculum,
          BasedOn = newlist[i].BasedOn,
          Qualification = newlist[i].Qualification,
          TrainingPeriod = newlist[i].TrainingPeriod,
          NameCathedra = newlist[i].Specialization.Cathedra.cathedraName,
          NameEducationalDegree = newlist[i].EducationalDegree.degreeName,
          NameFaculty = newlist[i].Specialization.Cathedra.Faculty.facultyName,
```

```
            NameForm_of_education = newlist[i].Form_of_education.Name,
            NameSpecialization = newlist[i].Specialization.nameSpecialization,
            NameSpesialty = newlist[i].Specialization.DirectorySpeciality.specialityName,
            NameKnowledgeArea = newlist[i].Specialization.DirectorySpeciality.KnowledgeArea.areaName
        });
    }
    return newlistdto;
}

public CurriculumDTO FindCurriculum(string name)
{
    Curriculum item = Database.RCurriculum.Get(name);
    CurriculumDTO DTO = null;

    if (item != null)
    {
        DTO = new CurriculumDTO();
        DTO.Id_Curriculum = item.IDCurriculum;
        DTO.BasedOn = item.BasedOn;
        DTO.Qualification = item.Qualification;
        DTO.TrainingPeriod = item.TrainingPeriod;
        DTO.NameCathedra = item.Specialization.Cathedra.cathedraName;
        DTO.NameEducationalDegree = item.EducationalDegree.degreeName;
        DTO.NameFaculty = item.Specialization.Cathedra.Faculty.facultyName;
        DTO.NameForm_of_education = item.Form_of_education.Name;
        DTO.NameSpecialization = item.Specialization.nameSpecialization;
        DTO.NameSpesialty = item.Specialization.DirectorySpeciality.specialityName;
        DTO.NameKnowledgeArea = item.Specialization.DirectorySpeciality.KnowledgeArea.areaName;
    }
    return DTO;
}

public async Task<OperationDetails> Create(CurriculumDTO DTO)
{
    Curriculum newItem = new Curriculum()
    {
        BasedOn = DTO.BasedOn,
        Qualification = DTO.Qualification,
        TrainingPeriod = DTO.TrainingPeriod,
        ID_EducationalDegree = Database.REducationalDegree.GetAll().Where(x => x.degreeName ==
                DTO.NameEducationalDegree).SingleOrDefault().IdDegree,
```

```
            ID_Form_of_education = Database.RFormEducation.GetAll().Where(x => x.Name ==
                DTO.NameForm_of_education).SingleOrDefault().IdFormEducation,
            ID_Specialization = Database.RSpecialization.GetAll().Where(x => x.nameSpecialization ==
                DTO.NameSpecialization).SingleOrDefault().ID_Specialization
        };

        Database.RCurriculum.Create(newItem);
        await Database.Save();
        return new OperationDetails(true, "Registration success", "");
    }
}

public class CurriculumDepartmentController : Controller
{
    public IActionResult VHome()
    {
        return View("~/Views/CurriculumDepartment/Home.cshtml");
    }

    private readonly ICycleService cycle;
    private readonly ISpecializationService specialization;
    private readonly IFormEducationService formEducation;
    private readonly IEducationalDegreeService educationalDegree;
    private readonly IPlanEducationalProcessService planEducationalProcess;
    private readonly IFormDeliveryService formDelivery;
    private readonly ICurriculumService curriculum;
    private readonly IUnderCycleService underCycle;
    private readonly IKnowledgeAreaService knowledgeArea;
    private readonly IFacultyService faculty;
    private readonly ICathedraService cathedra;
    private readonly ISpecialityService speciality;

    public CurriculumDepartmentController(ICycleService servCy, ISpecializationService servS, IFormEducationService servFE,
        IEducationalDegreeService servED, IPlanEducationalProcessService servPE, IFormDeliveryService servFD, ICurriculumService servCu,
            IUnderCycleService servUC, IKnowledgeAreaService servK, IFacultyService servF, ICathedraService servCa, ISpecialityService servSp)
    {
        cycle = servCy;
        specialization = servS;
        formEducation = servFE;
```

```
    educationalDegree = servED;
    planEducationalProcess = servPE;
    formDelivery = servFD;
    curriculum = servCu;
    underCycle = servUC;
    knowledgeArea = servK;
    faculty = servF;
    cathedra = servCa;
    speciality = servSp;
}

#region Curriculum
public IActionResult GetCurriculum()
{
    CurriculumModel newModel = new CurriculumModel();
    newModel.curriculum = curriculum.GetAll().ToList();
    return View("~/Views/CurriculumDepartment/Curriculum/Curriculum.cshtml", newModel);
}

public IActionResult GetInfoCurriculum(string id)
{
    CurriculumInfoModel newModel = new CurriculumInfoModel();
    newModel.curriculum = curriculum.FindCurriculum(id);
    newModel.newList = planEducationalProcess.GetAllOneCurriculun(Convert.ToInt32(id)).ToList();
    return View("~/Views/CurriculumDepartment/Curriculum/CurriculumnInfo.cshtml", newModel);
}

public async Task<IActionResult> AddNewCurriculum(AddCurriculumModel model)
{
    CurriculumDTO newItem = new CurriculumDTO()
    {
        Qualification = model.Qualification,
        NameSpecialization = model.nameSpecialization,
        NameEducationalDegree = model.nameEducationalDegree,
        NameForm_of_education = model.nameForm_of_education,
        TrainingPeriod = model.TrainingPeriod,
        BasedOn = model.BasedOn
    };

    await curriculum.Create(newItem);
    return RedirectToAction("GetCurriculum", "CurriculumDepartment");
}
```

```
public IActionResult GetAddCurriculum()
{
   AddCurriculumModel newModel = new AddCurriculumModel();
   newModel.newListForm_of_education = formEducation.GetAll().ToList().Select(x => new SelectListItem { Text =
            x.FormEducationName, Value = x.FormEducationName }).ToList();
   newModel.newListEducationalDegree = educationalDegree.GetAll().ToList().Select(x => new SelectListItem { Text =
            x.EducationalDegreeName, Value = x.EducationalDegreeName }).ToList();
   newModel.newKnowledgeArea = knowledgeArea.GetAll().ToList().Select(x => new SelectListItem { Text =
            x.areaCode + "  " + x.areaName, Value = x.areaName }).ToList();
   newModel.newFaculty = faculty.GetAll().ToList().Select(x => new SelectListItem { Text = x.facultyName, Value =
            x.facultyName }).ToList();

   return View("~/Views/CurriculumDepartment/Curriculum/AddCurriculum.cshtml", newModel);
}

[HttpGet]
public ActionResult GetItemsFaculty(string id)
{
   List<CathedraDTO> asd = cathedra.GetAll().Where(c => c.facultyName == id).ToList();
   AddCurriculumModel newModel = new AddCurriculumModel();
   newModel.newListCathedra = asd.Select(x => new SelectListItem() { Text = x.cathedraName, Value = x.cathedraName
            }).ToList();
   newModel.newListSpecialization = null;
   return PartialView("~/Views/CurriculumDepartment/Curriculum/GetItemsFaculty.cshtml", newModel);
}

[HttpGet]
public ActionResult GetItemsSpeciality(string id)
{
   List<SpecialityDTO> asd = speciality.GetAll().Where(c => c.KnowledgeAreaName == id).ToList();
   AddCurriculumModel newModel = new AddCurriculumModel();
   newModel.newListSpeciality = asd.Select(x => new SelectListItem() { Text = x.Code_Speciality +" "+
            x.Name_Speciality, Value = x.Name_Speciality }).ToList();
   newModel.newListSpecialization = null;
   return PartialView("~/Views/CurriculumDepartment/Curriculum/GetItemsSpeciality.cshtml", newModel);
}

[HttpGet]
public ActionResult GetItemsSpecialization(string id)
{
   string[] name = id.Split(new char[] { '&' });
```

```
    string nameSpeciality = name[0];
    string nameCathedra = name[1];

    List<SpecializationDTO> asd = specialization.GetAll().Where(c => c.Speciality == nameSpeciality && c.nameCathedra
            == nameCathedra).ToList();
    AddCurriculumModel newModel = new AddCurriculumModel();
    newModel.newListSpecialization = asd.Select(x => new SelectListItem() { Text = x.nameSpecialization, Value =
            x.nameSpecialization }).ToList();
    return PartialView("~/Views/CurriculumDepartment/Curriculum/GetItemsSpecialization.cshtml", newModel);
}

public ActionResult ExportExelCurriculumn()
{
    using (XLWorkbook workbook = new XLWorkbook(XLEventTracking.Disabled))
    {
        var worksheet = workbook.Worksheets.Add("Навчальний план");

        worksheet.Cell("A1").Value = "1";
        worksheet.Row(1).Style.Font.Bold = true;

        using (var stream = new MemoryStream())
        {
            workbook.SaveAs(stream);
            stream.Flush();

            return new FileContentResult(stream.ToArray(), "aplication/vnd.openxmlformats-
                officedocument.spreadsheetml.sheet")
            {
                FileDownloadName = $"brands_{DateTime.UtcNow.ToShortDateString()}.xlsx"
            };
        }
    }
}

public async Task<IActionResult> AddNewPlanEducationalProcess(AddPlanEducationalProcessModel model)
{
    PlanEducationalProcessDTO newItem = new PlanEducationalProcessDTO()
    {
        NameDiscipline = model.NameDiscipline,
        ID_Curriculm = model.ID_Curriculm,
        Course = model.Course,
        Projects = model.Projects,
```

```
        LaboratoryHours = model.LaboratoryHours,
        PracticesHours = model.PracticesHours,
        IndependentWorkHours = model.IndependentWorkHours,
        LectureHours = model.LectureHours,
        Semester = model.Semester,
        nameFormDelivery = model.nameFormDelivery,
        nameUnderCycle = model.nameUnderCycle,
    };

    await planEducationalProcess.Create(newItem);
    return RedirectToAction("GetAddPlanEducationalProcess", "CurriculumDepartment");
}

public IActionResult GetAddPlanEducationalProcess()
{
    AddPlanEducationalProcessModel newModel = new AddPlanEducationalProcessModel();
    newModel.newListFormDelivery = formDelivery.GetAll().ToList().Select(x => new SelectListItem { Text =
            x.FormDeliveryName, Value = x.FormDeliveryName }).ToList();
    newModel.newListIdCurriculm = curriculum.GetAll().ToList().Select(x => new SelectListItem { Text =
            x.Id_Curriculum.ToString(), Value = x.Id_Curriculum.ToString() }).ToList();
    newModel.newListCycle = cycle.GetAll().ToList().Select(x => new SelectListItem { Text = x.CycleName, Value =
            x.CycleName}).ToList();

    return View("~/Views/CurriculumDepartment/Curriculum/AddPlanEducationalProcess.cshtml", newModel);
}

[HttpGet]
public ActionResult GetItemsUnderCycle(string id)
{
    List<UnderCycleDTO> asd = underCycle.GetAll().Where(c => c.CycleName == id).ToList();
    AddPlanEducationalProcessModel newModel = new AddPlanEducationalProcessModel();
    newModel.newListUnderCycle = asd.Select(x => new SelectListItem() { Text = x.UnderCycleName, Value =
            x.UnderCycleName }).ToList();
    return PartialView("~/Views/CurriculumDepartment/Curriculum/GetItemsUnderCycle.cshtml", newModel);
}

#endregion
```

ДОДАТОК В

Модуль розробки навчальних планів інформаційної системи автоматизованої підтримки навчальної діяльності кафедри

Опис програмного модулю

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61106_20Б

Аркушів 8

Київ – 2020

АНОТАЦІЯ

Додаток являє собою систему для створення навчальних планів. Додаток має роль – керівник по навчальних планах кафедри. У функціонал входить: можливість створення всієї необхідної інформації для навчального плану, створення самого навчального плану, перегляд всієї інформації у таблицях.

Модуль було розроблено з використанням мови програмування C#, фреймворку Entity Framework, Asp.Net Core, технології MVC, Ms Sql, середовища розробки Visual Studio Code.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

Дану програмну систему було розроблено з використанням середовища Visual Studio 2019 мовою програмування C# з використанням .Net Framework Core та технологій MVC, Entity Framework.

Програма призначена для автоматизованого створення навчальних планів кафедри, можливості переглядати всі дані у вигляді таблиці.

До основних переваг програми можна віднести швидкість роботи та відносно простий для користувача інтерфейс.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

При створенні програмного продукту була поставлена задача створення системи для автоматизованої підтримки навчальної діяльності кафедри. Для зберігання даних була створена база даних. За допомогою технології Entity, базу даних можна легко розширювати і змінювати. Програмний продукт містить декілька різних типів методів, які виконують наступні функції:

- швидка взаємодія з базою даних;
- створення навчальних планів;
- заповнення інформації необхідної для коректної роботи навчальних планів;
- можливість перегляду всієї інформації.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Створена програма складається з декількох частин:

- бази даних (створення структури бази даних);
- бізнес логіки (проміжний модуль, який працює з базою даних і представленням);
- представлення (робота користувача з представленням у вигляді WEB-форм).

Робота програми побудована таким чином, що відкривши Projekt.sln користувач відкриває форму авторизації. Авторизувавшись перед користувачем постає ряд можливостей:

- заповнити всю необхідну інформацію, необхідну для створення навчального плану;
- перегляд всієї інформації у відповідних таблицях;
- створення навчального плану.

ВИКОРИСТУВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для створення даної програми було використано середовище розробки Visual Studio 2019 та мова програмування C# з використанням Asp.Net Core та технологій MVC, Entity Framework.

Для зберігання великих об’ємів даних була створена база даних Ms Sql.

Для запуску даної програми користувачу необхідно мати доступ до мережі Інтернет.

ВХІДНІ ТА ВИХІДНІ ДАНІ

Вхідними даними є:

– дані про методичні матеріали та навчальні дисципліни.

Вихідними даними є:

– створені навчальні плани.

ДОДАТОК Г

Модуль розробки навчальних планів інформаційної системи автоматизованої підтримки навчальної діяльності кафедри

Апробація

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61106_20Б

Аркушів 4

Київ – 2020

Було подано тези на конференцію «Матеріали VIII Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених з автоматичного управління присвячена Дню космонавтики». Тези надруковано в першій секції “Автоматизоване управління технологічними процесами” на сторінці 9. Текст тез наведено нижче.

Міністерство освіти і науки України

Херсонський національний технічний університет

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Вінницький національний медичний університет
ім. М. І. Пирогова

Луцький національний технічний університет

Вінницький національний технічний університет

Кременчуцький національний технічний університет
ім. Михайла Остроградського

Сумський державний університет

Херсонський державний аграрно-економічний університет

Матеріали VIII Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених з автоматичного управління

присвячена Дню космонавтики

08 – 10 квітня 2020р.
Херсон

ЗМІСТ

СЕКЦІЯ «АВТОМАТИЗОВАНЕ УПРАВЛІННЯ ТЕХНОЛОГІЧНИМИ ПРОЦЕСАМИ»

СЕКЦІЯ «МОДЕЛЮВАННЯ ТА ОПТИМІЗАЦІЯ СИСТЕМ УПРАВЛІННЯ»

УДК 004.414.38

О.М. Белень, А.В. Дмитрук, О.М. Шушура
НТУУ «КПІ ім. Ігоря Сікорського»
alexandrbelen@gmail.com

АВТОМАТИЗАЦІЯ УПРАВЛІННЯ НАВЧАЛЬНОЮ ДІЯЛЬНІСТЮ УНІВЕРСИТЕТУ НА БАЗІ ASP.NET Core 3.0 ТА ANGULAR 9

Інформаційні технології стали невід'ємною частиною сучасного життя. Вони назавжди змінили світ бізнесу, культури, освіти, виробництва. Використання інформаційних технологій дозволило суттєво збільшити ефективність навчального процесу. Водночас постала проблема необхідності обробки великої кількості інформації, а отже, створення таких систем, що будуть полегшувати управління навчальним процесом. Такий напрям, як інформатизація освіти, має місце в усіх без винятку національних програмах руху до інформаційного суспільства.

Навчальний процес у вищому навчальному закладі достатньо важкий, і його ефективна організація потребує значних зусиль. У сучасних університетах автоматизація відбувається у двох основних взаємопов'язаних напрямках.

Перший напрям – автоматизація освітнього процесу, використання сучасних інформаційних технологій для модернізації педагогічного процесу (дистанційне навчання, електронне та всепроникне навчання).

Другий напрям – автоматизація системи університетського менеджменту шляхом розробки інформаційних технологій для бізнес-процесів сучасного університету.

В даній роботі розглядається комплексний підхід, який враховує обидва вказані напрямки. Огляд існуючих інформаційних систем в цій галузі дозволив виділити наступні принципи їх побудови:

- використання даних із загального сховища даних (інтегрована БД) з розмежуванням прав доступу на рівні користувачів і окремих додатків (окремих показників);
- використання загальних довідників;
- обмін інформацією між підсистемами на основі єдиного інформаційного середовища.

Проектування інтегрованої інформаційної системи повинне розглядатися крізь призму основних функцій управління, основних напрямів діяльності університету та його внутрішньої структури, що формується залежно від виконуваних завдань. Система повинна справлятися із значним навантаженням, одночасно обробляти запити користувачів та мати зручний і інтуїтивно зрозумілий інтерфейс.

Для створення клієнтської частини порталу обрані такі технології як Angular 9–фреймовк на основі паттерна MVC,інтеграція з ASP.NET Core 3.0, запити та обробка даних.

Основою серверної частини є крос-платформна система ASP.NET Core 3.0. Вибір даних технологій зумовило те, що дана система є складною в проектуванні та підтримці, а даний фреймворк призначений саме для таких випадків.

Використання наведених сучасних технологій розробки інформаційних систем дозволяє врахувати всі суттєві вимоги до управління навчальним процесом та забезпечити надійність і зручність застосування програмного продукту.

ЛІТЕРАТУРА:

1. Методологічні основи створення впровадження і розвитку інтегрованої інформаційної системи управління університетом. 2015 URL: https://essuir.sumdu.edu.ua/bitstream-download/123456789/47881/3/integrated_management_system_ukr.pdf.
2. Бибо Б. jQuery. Подробное руководство по продвинутому JavaScript. М.: Символ-Плюс, 2011.
3. ASP.NET Core. URL: https://docs.microsoft.com/en-us/aspnet/?view=aspnetcore-3.1.